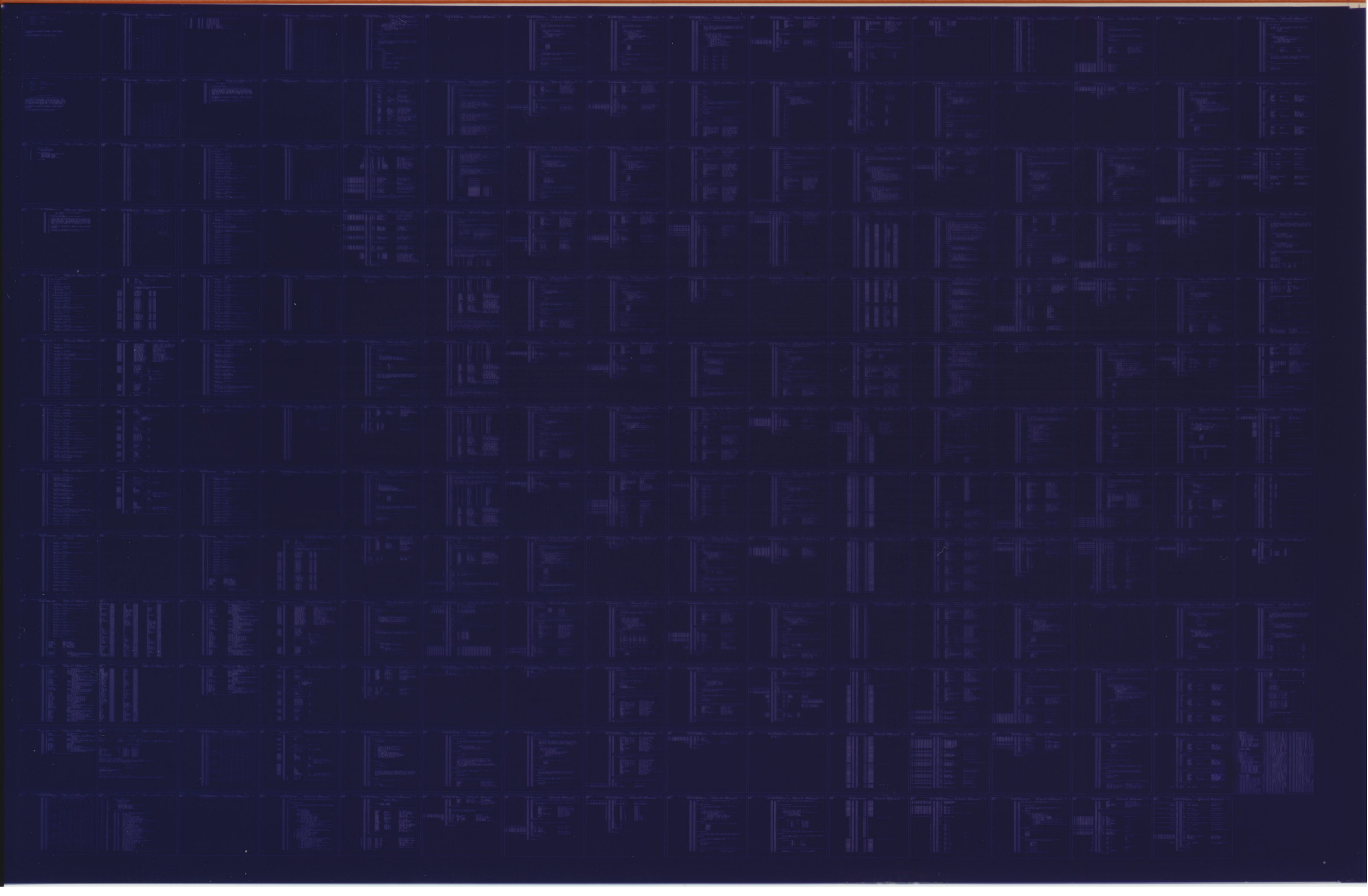
ECKAB-9.2

11/750 DPM MICRO DIAG

EP-ECKAB-DL-9.2
1 OF 7 JUN 1987
COPYRIGHT© 1980-87

digital
MADE IN USA

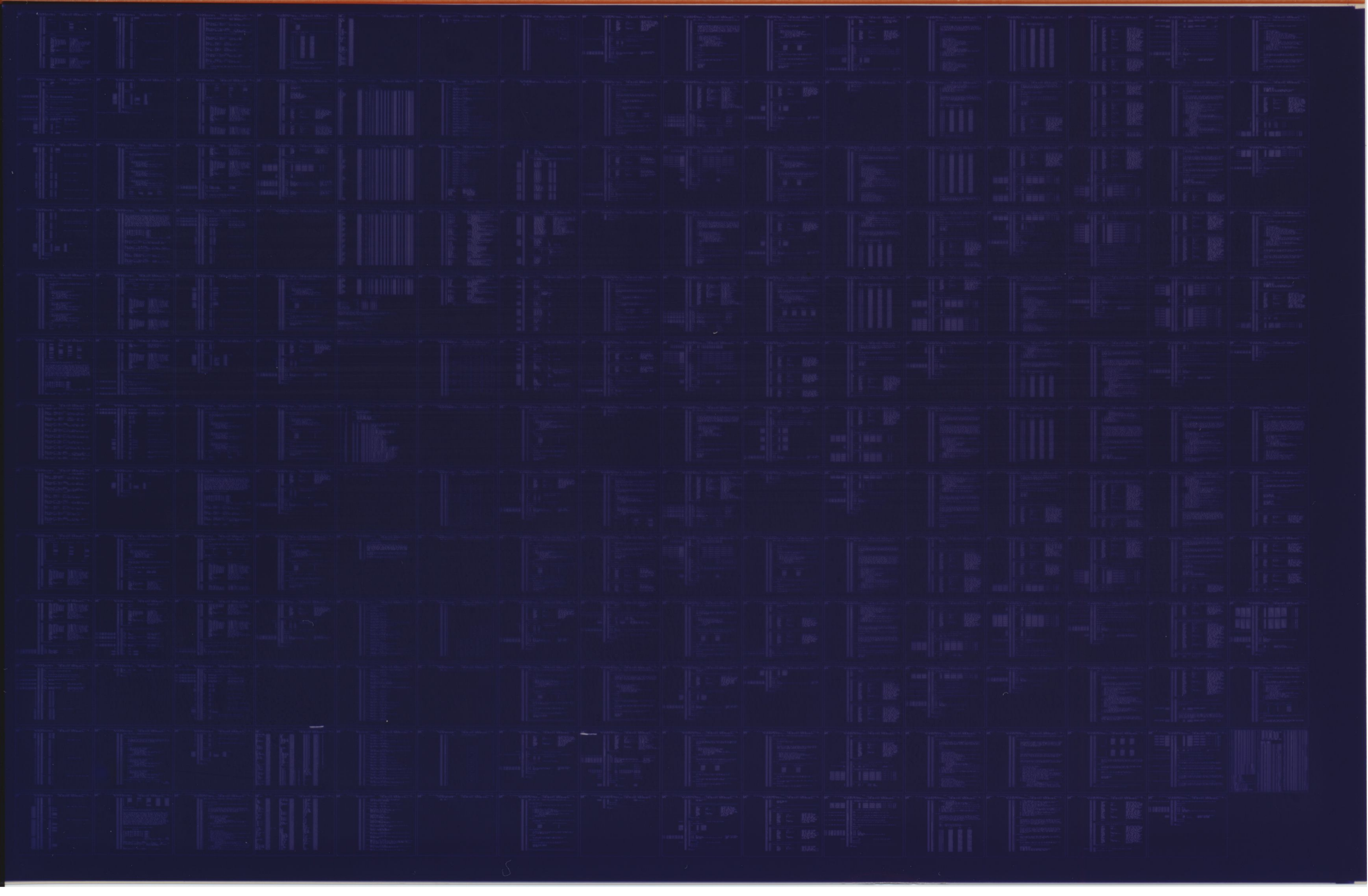


ECKAB-9.2

11/750 DPM MICRO DIAG

EP-ECKAB-DL-9.2
2 OF 7 JUN 1987
COPYRIGHT© 1980-87

digital
MADE IN USA



ECKAB-9.2

11/750 DPM MICRO DIAG

EP-ECKAB-DL-9.2
3 OF 7 JUN 1987
COPYRIGHT© 1980-87

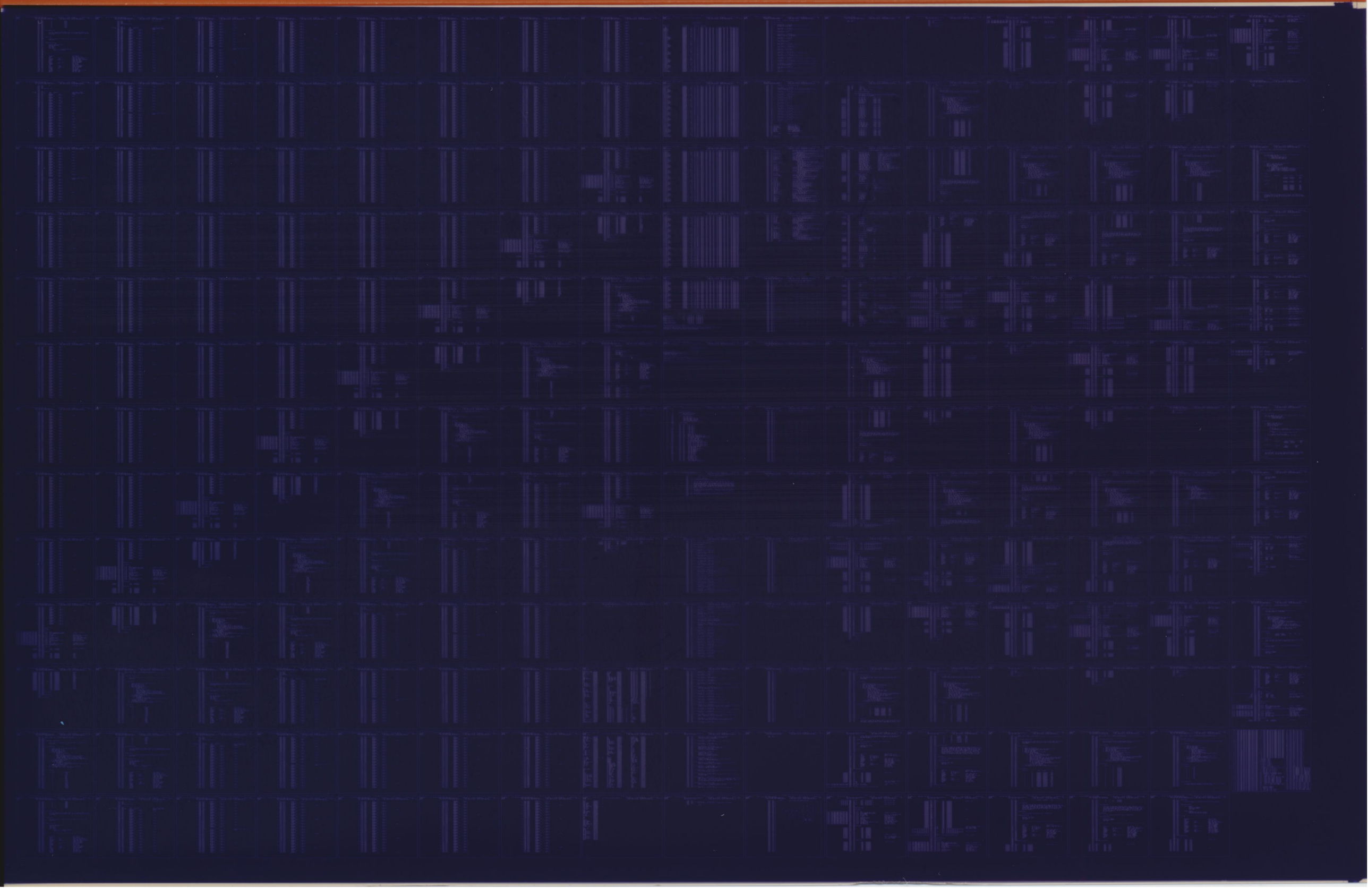
digital
MADE IN USA

ECKAB-9.2

11/750 DPM MICRO DIAG

EP-ECKAB-DL-9.2
4 OF 7 JUN 1987
COPYRIGHT© 1980-87

digital
MADE IN USA



ECKAB-9.2

11/750 DPM MICRO DIAG

EP-ECKAB-DL-9.2
5 OF 7 JUN 1987
COPYRIGHT© 1980-87

digital
MADE IN USA

ECKAB-9.2

11/750 DPM MICRO DIAG

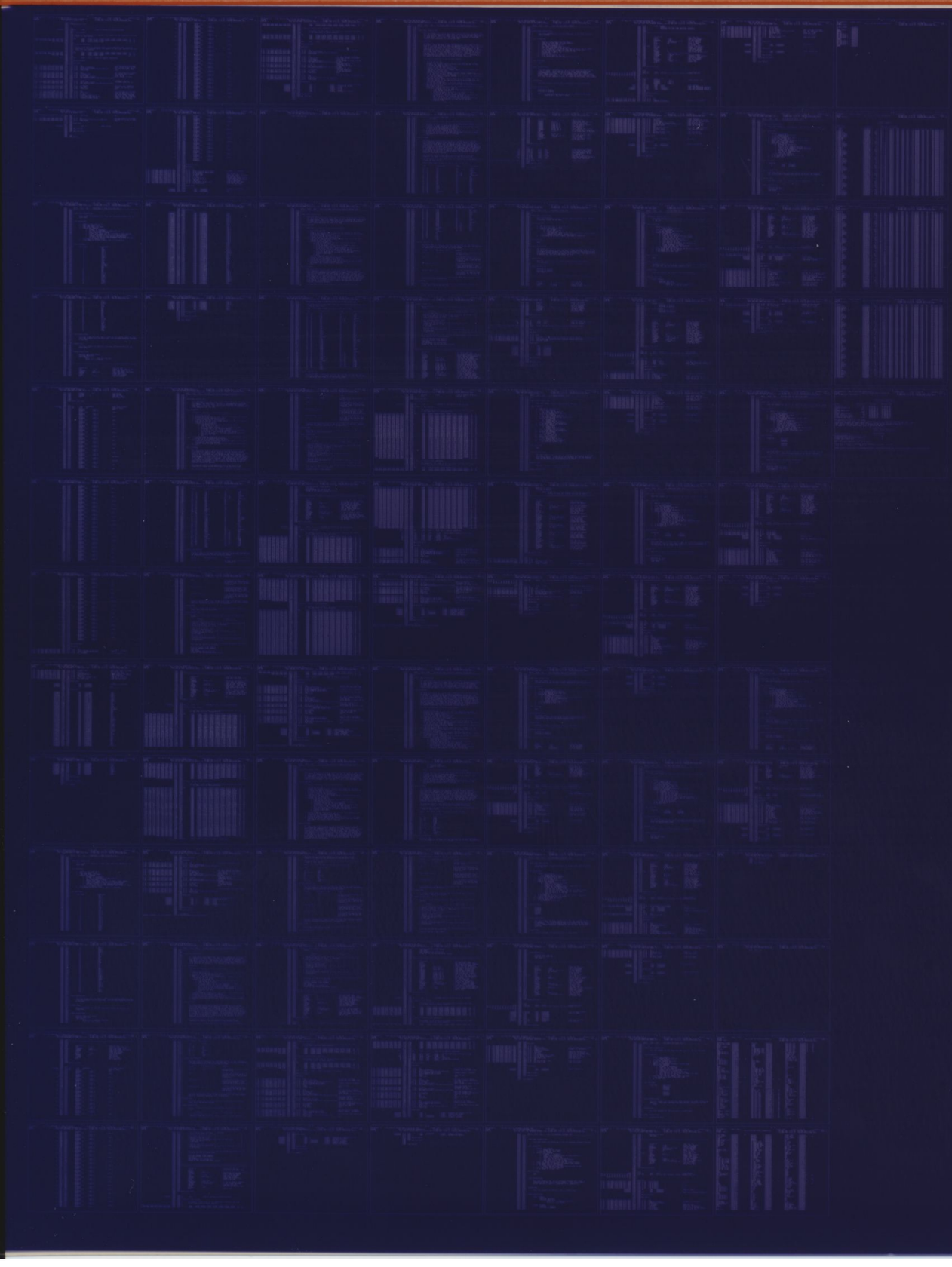
EP-ECKAB-DL-9.2
6 OF 7 JUN 1987
COPYRIGHT© 1980-87

digital
MADE IN USA

ECKAB-9.2

11/750 DPM MICRO DIAG

EP-ECKAB-DL-9.2
7 OF 7 JUN 1987
COPYRIGHT© 1980-87
MADE IN USA



PRODUCT ID: ZZ-ECKAB-9.2

PRODUCT TITLE: 11/750 DPM Micro Diagnostic

DECO/DEPO: 9.2

DATE: 18-MAR-1987

DEPARTMENT: PEDM

THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

PRODUCT ID: ZZ-ECKAB-9.2

PRODUCT TITLE: 11/750 DPM Micro Diagnostic

DECO/DEPO: 9.2

DATE: 18-MAR-1987

DEPARTMENT: PEDM

COPYRIGHT (C) 1980,1987

DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.

THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

(1) 24 "DPM Revision History"
(1) 1 ""
(1) 3 "DIAGNOSTIC MICROCODE MACROS"
(1) 98 " RDM Control File Macros"
(1) 109 " Diagnostic Macros"
(1) 194 ""
(1) 195 "MODULE GATE ARRAY MAPS"
(1) 197 " L0001 MULTIPLIER AND GATE ARRAY LOCATION"
(1) 254 " L0002 GATE ARRAY LOCATION"
(1) 311 " L0003 GATE ARRAY LOCATION"
(1) 369 " L0004 GATE ARRAY LOCATION"
(1) 427 " L0011 & L0016 GATE ARRAY LOCATION"
(1) 485 ""
(1) 490 "EQUATED SYMBOLS
(1) 493 ""

```
0000     1          .TITLE  ECKAB VAX-11/750 DPM MICRO DIAGNOSTICS
0000     2          .IDENT  /V09.02/
0000     3 ;
0000     4 ; COPYRIGHT (C) 1980,1983
0000     5 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     6 ;
0000     7 ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE  ONLY ON A SINGLE
0000     8 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000     9 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    10 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    11 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    12 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    13 ; REMAIN IN DEC.
0000    14 ;
0000    15 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    16 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    17 ; CORPORATION.
0000    18 ;
0000    19 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    20 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    21 ;
0000    22 ;
```

```
0000     24          .SBTTL  "DPM Revision History"
0000     25 ;++
0000     26 ; DPM MICRO-DIAGNOSTIC PACKAGE
0000     27 ;
0000     28 ; REVISION HISTORY
0000     29 ;
0000     30 ; 00-01 Tom Groetzinger 4-FEB-1980
0000     31 ;       Started file.
0000     32 ;
0000     33 ; 00-02 Dave Spain      27-FEB-1980
0000     34 ;       Added CS BUS INTEGRITY test.
0000     35 ;
0000     36 ; 00-03 Tom Groetzinger 12-MAR-1980
0000     37 ;       Added five tests.
0000     38 ;
0000     39 ; 00-04 Tom Groetzinger 17-MAR-1980
0000     40 ;       Added SRK,SRM callouts to some tests.
0000     41 ;
0000     42 ; 00-05 Tom Groetzinger 17-MAR-1980
0000     43 ;       Improved tests 7 and 16 to increase coverage due to Westboro feedback
0000     44 ;
0000     45 ; 00-06 Tom Groetzinger 17-MAR-1980
0000     46 ;       Debugged test 16 dueu to major rewrite
0000     47 ;
0000     48 ; 00-07 Tom Groetzinger 17-MAR-1980
0000     49 ;       Added 9 Super Rotator tests
0000     50 ;
0000     51 ; 01-08 Dave Spain       4-APR-80
0000     52 ;       Added 4 Micro-sequencer tests for offical release.
0000     53 ;
0000     54 ; 01-09 Bill Landry     4-APR-1980
0000     55 ;       Added the XB, PREFETCH and IRD tests from ECKAC.SRC so
0000     56 ;       manufacturing can run one tape for DPM.
0000     57 ;
0000     58 ; 00-10 Bill Landry     4-APR-1980
0000     59 ;       Let's save version 1.0 for our first SDC release.
0000     60 ;
0000     61 ; 00-11 Tom Groetzinger 7-APR-1980
0000     62 ;       Added 2 ALK Tests
0000     63 ;
0000     64 ; 00-12 Tom Groetzinger 8-APR-1980
0000     65 ;       Deleted last 2 D Register outputs from ALK Test #6
0000     66 ;
0000     67 ; 00-13 Tom Groetzinger 14-APR-1980
0000     68 ;       Added D Register output to ALK test #6
0000     69 ;
0000     70 ; 00-14 Dave Spain      25-APR-1980
0000     71 ;       Changed mask on the micro-sequencer tests to ignore bit 13 of the
0000     72 ;       Control Store address.
0000     73 ;
0000     74 ; 00-15 Dave Spain      28-APR-1980
0000     75 ;       Fixed bug in ALK Tests 6 and 7 by replacing VSIZE/1 micro order with a
0000     76 ;       SIZE[LONG] macro in micro words 6 and 3 respectively.
0000     77 ;
0000     78 ; 00-16 Dave Spain      29-APR-1980
0000     79 ;       Changed mask on CS address for Instruction Address Decode test from
0000     80 ;       3FFF to 1FFF to ignore bit 13 of Control Store address.
```

```
0000     81 ;
0000     82 ; 00-17 Dave Spain      30-APR-1980
0000     83 ;       Added CCS module to error callout for Control Store Bus Integrity test.
0000     84 ;
0000     85 ; 00-18 Dave Spain       9-MAY-1980
0000     86 ;       Added SRM gate array to error callout routine in RTMP0 and RTEMP
0000     87 ;       Sratchpad tests.
0000     88 ;
0000     89 ; 00-19 Bill Landry     10-JUN-1980
0000     90 ;       Added RDM tests to beginning of tape
0000     91 ;
0000     92 ; 01-00 Bill Landry/Dave Spain  13-JUN-1980
0000     93 ;       Added macro file to title section and compatibility mode IRD tests
0000     94 ;       then submitted to RELEASE ENGINERRING.
0000     95 ;
0000     96 ; 01-01 Dave Spain      19-JUN-1980
0000     97 ;       Fixed prefetch incrementer test to clear out cache data after every
0000     98 ;       prefetch. This will prevent the test from mistaking bad addresses from
0000     99 ;       good ones.
0000    100 ;
0000    101 ; 01-02 Dave Spain       2-JUL-1980
0000    102 ;       Added CM Second IRDX tests to file.
0000    103 ;
0000    104 ; 01-03 Dave Spain      18-AUG-1980
0000    105 ;       Added CM branch test to file.
0000    106 ;
0000    107 ; 01-04 Dave Spain      10-SEPT-1980
0000    108 ;       Fixed "DEFAULT MTEMP0 TEST"
0000    109 ;
0000    110 ; 01-05 Bill Landry     11-SEPT-1980
0000    111 ;       Revised RDM to CMI tests
0000    112 ;
0000    113 ; 02-00 Dave Spain      28-SEPT-1980
0000    114 ;       Added rotator tests to file and submitted to SDC.
0000    115 ;
0000    116 ; 02-01 Dave Spain      28-OCT-1980
0000    117 ;       Added QV flag to tests to be run by manufacturing.
0000    118 ;
0000    119 ; 02-02 Bill Landry     10-NOV-1980
0000    120 ;       Added tests for TOK and SAC gate arrays.
0000    121 ;
0000    122 ; 02-03 Bill Landry     13-NOV-1980
0000    123 ;       Added test of REI CHECK
0000    124 ;
0000    125 ; 03-00 Dave Spain      17-NOV-1980
0000    126 ;       Added rotator tests 23-28 and released to SDC.
0000    127 ;
0000    128 ; 03-01 Dave Spain      10-DEC-1980
0000    129 ;       Added rotator tests 29-42.
0000    130 ;
0000    131 ; 03-02 Bill Landry     11-DEC-1980
0000    132 ;       Fixed bug in CS ADDRESS test
0000    133 ;
0000    134 ; 03-03 Dave Spain      12-DEC-1980
0000    135 ;       Enhanced Rotator Test 1 to call out ALP slices.
0000    136 ;
0000    137 ; 03-04 Bill Landry     16-DEC-1980
```

```
0000    138 ;       Added DSIZE ROM tests
0000    139 ;
0000    140 ; 03-05 Bill Landry     31-DEC-1980
0000    141 ;       Last great act of 80. Separated DPM and RDM tests into two files.
0000    142 ;
0000    143 ; 04-00 Bill Landry     5-JAN-1981
0000    144 ;       Submitted to RELEASE ENGINERRING
0000    145 ;
0000    146 ; 04-01 Bill Landry     14-JAN-1981
0000    147 ;       Revised INTERVAL TIMER INTERRUPT TEST to fix bug caused by INT PEND
0000    148 ;       ECO
0000    149 ;
0000    150 ; 04-02 Dave Spain      16-JAN-1981
0000    151 ;       Fixed error callout in CS Parity Error, XB, and Prefetch Incrementer
0000    152 ;       tests and added seven new rotator tests.
0000    153 ;
0000    154 ; 04-03 Bill Landry     4-FEB-1981
0000    155 ;       Added tests for CCC gate array
0000    156 ;
0000    157 ; 04-04 Dave Spain      9-FEB-1981
0000    158 ;       Changed the names of rotator tests 23-33.
0000    159 ;
0000    160 ; 04-05 Bill Landry     24-FEB-1981
0000    161 ;       Added tests for DPM discrete branching logic.
0000    162 ;
0000    163 ; 04-06 Bill Landry     27-FEB-1981
0000    164 ;       Added tests for branching logic in PHB gate array.
0000    165 ;
0000    166 ; 05-00 Dave Spain       1-MAR-1981
0000    167 ;       Added 5 SRK status rotator tests and submitted to SDC.
0000    168 ;
0000    169 ; 05-01 Dave Spain      25-MAR-1981
0000    170 ;       Fixed a loop on test bug in the Alu Functions Test by adding a LDFIELD
0000    171 ;       pseudo-op that reinitializes the ROT field in Subtest 1.
0000    172 ;
0000    173 ; 05-02 Bill Landry     25-MAR-1981
0000    174 ;       Fixed bug in BUT/NO.FPA test.  I was branching on the wrong conditions.
0000    175 ;
0000    176 ; 05-03 Dave Spain      14-APR-1981
0000    177 ;       Cleaned up documentation for R Scratchpads Write All Cells Test.
0000    178 ;
0000    179 ; 06-00 Bill Landry     2-MAY-1981
0000    180 ;       Added SPA branching, CON registers, TU58 registers, and CCS PARITY
0000    181 ;       tests.  Submitted to Release Engineering.
0000    182 ;
0000    183 ; 06-01 Dave Spain      17-JUN-1981
0000    184 ;       Replaced the old IRD1 Instruction Address Decode test with two new
0000    185 ;       tests that toggle the PSL's FPD bit and can handle an FPA.
0000    186 ;
0000    187 ; 06-02 Bill Landry     19-AUG-1981
0000    188 ;       Fixed bugs created by introduction of FPA option.
0000    189 ;       Replaced all references to COMET with the word CPU.
0000    190 ;       Fixed typing error in vbus data of CPU CLOCK TEST.
0000    191 ;       Submitted to RELEASE ENGINEERING.
0000    192 ;
0000    193 ; 06-03 Bill Landry     24-SEPT-1981
0000    194 ;       Fixed potential clock problem with FPA in Z BIT TEST 1.
```

```
0000   195 ;
0000   196 ; 06-04 Bill Landry     9-OCT-1981
0000   197 ;       Added changes for ECO L0002-TW002
0000   198 ;
0000   199 ; 07-00 Bill Landry     31-OCT-1981
0000   200 ;       Removed changes for ECO L0002-TW002.  ECO is on hold.
0000   201 ;       Added tests for FPA to CPU interface logic
0000   202 ;       Moved WCS test to MIC tape
0000   203 ;       Submitted to RELEASE ENGINEERING.
0000   204 ;
0000   205 ; 07-01 Bill Landry     13-NOV-1981
0000   206 ;       Stopped release of 7.0
0000   207 ;       Added FPA TRAP LOGIC TEST
0000   208 ;       Enhanced error messages for FPA
0000   209 ;       Resubmitted to RELEASE ENGINEERING
0000   210 ;
0000   211 ; 07-02 Bill Landry     20-NOV-1981
0000   212 ;       Stopped release of 7.1
0000   213 ;       Enhanced FPA ENABLE/DISABLE TEST to check for MSEQ INIT
0000   214 ;       Resubmitted to RELEASE ENGINEERING
0000   215 ;
0000   216 ; 07-03 Dave Shull      26-Mar-1982
0000   217 ;       Added the ERRLOOP pseudo to the Rotator Extract tests which were
0000   218 ;       missing these and couldn't be looped on.
0000   219 ;
0000   220 ; 07-04 Dave Shull      05-Apr-1982
0000   221 ;       Removed all BEGINSA and ENDSA pseudo's.
0000   222 ;       Submitted to RELEASE ENGINEERING
0000   223 ;
0000   224 ; 07-05 Dave Shull      21-May-1982
0000   225 ;       Add a subtest to the Micro Sequencer Stack Test that will test bits
0000   226 ;       11 and 12 of the micro stack.
0000   227 ;
0000   228 ; 07-06 Dave Shull      24-May-1982
0000   229 ;       Changed documentation in the CM Odd Address BUT Test to reflect the
0000   230 ;       ECO L0002-TW002
0000   231 ;
0000   232 ; 08-00 Dave Shull      18-Jun-1982
0000   233 ;       Added 10 new tests.  Two of those tests are an addition to the existing
0000   234 ;       IRD1 rom tests.  These 2 new tests check the signal ROM OS INH H.
0000   235 ;       The other 8 new tests check the validity of the IRDX ROMs.  They check
0000   236 ;       both the address and the signal ROM OS INH H.
0000   237 ;
0000   238 ; 08-01 Dave Shull      22-Jun-1982
0000   239 ;       Added a subtest to the Register Backup Stack test that will check the
0000   240 ;       bit that records Increment/Decrement of the GPR.
0000   241 ;
0000   242 ; 09-00 Dave Shull      03-Nov-1982
0000   243 ;       Added two new Tests for coverage the new PCS (L0008) module.
0000   244 ;
0000   245 ; 09-01 Dave Smith      15-May-1983
0000   246 ;       Correct DCS in ALK test 3,6,7. Disabled ALP's on the WBUS.
0000   247 ;
0000   248 ; 09-02 Joe Zagarella   20-Jan-1987
0000   249 ;       Fixed bug introduced by L0003-YA module in tests CE-D3, and D6-DB
0000   250 ;--
0000     1         .SBTTL  ""
```

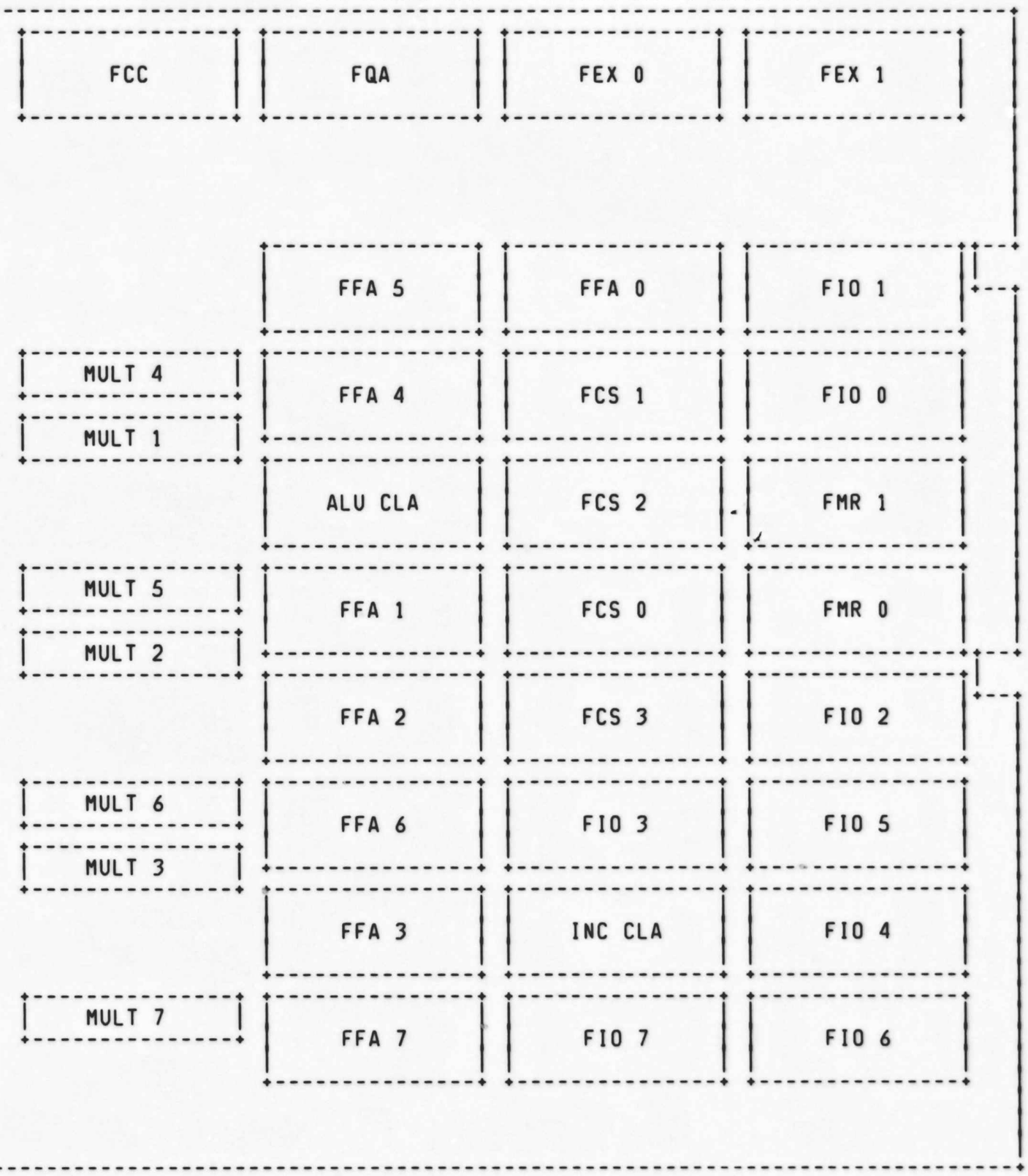
```
0000     3          .SBTTL  "DIAGNOSTIC MICROCODE MACROS"
0000     4 ;++
0000     5 ; Revision History
0000     6 ;
0000     7 ; 28    Dave Spain      22-Jun-82
0000     8 ;       Removed LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
0000     9 ;       Decided not to use them.
0000    10 ;
0000    11 ; 27    Dave Spain      15-Jun-82
0000    12 ;       Added LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
0000    13 ;
0000    14 ; 26    Dave Spain      19-May-82
0000    15 ;       Added M[]_FPA macro.
0000    16 ;
0000    17 ; 25    Rich Lewis      23-Feb-82
0000    18 ;       Added PSL_WB, M[]_R[].OR.NIB0(M), WB_FPA CCR, WB_FPA TCR, Q_R[].ASL.P,
0000    19 ;       RDM_M[].XZ.OR.R[], FPA_WB macros
0000    20 ;
0000    21 ; 24    Dave Spain      02-Feb-82
0000    22 ;       Added FPA_R[]+M[] macro.
0000    23 ;
0000    24 ; 23    Bill Landry     22-OCT-81
0000    25 ;       Added PSL_RDM macro
0000    26 ;
0000    27 ; 22    Dave Spain       5 Oct 81
0000    28 ;       Added RDM_FPA and FPA_M[] FPA_RDM macros.
0000    29 ;
0000    30 ; 21    Bill Landry     29 Sept 81
0000    31 ;       Added FPA macro, FPA_M[] FPA_R[].
0000    32 ;
0000    33 ; 20    Dave Spain      14 Aug 81
0000    34 ;       Added the TESTFPA [] macro.
0000    35 ;
0000    36 ; 19    Dave Spain      18 July 81
0000    37 ;       Added the following FPA macros, FPA_RDM and FPA_R[].
0000    38 ;
0000    39 ; 18    Dave Spain      24 Sept 80
0000    40 ;       Added RDM_Q macro. This macro uses the ALU.
0000    41 ;
0000    42 ; 17    Bill Landry     23 Sept 80
0000    43 ;       Added macro VA_VA-ZLIT0[] for testing memory in decending order.
0000    44 ;
0000    45 ; 16    Dave Spain      27 August 80
0000    46 ;       Changed RDM_Q macro to RDM_Q Q_D. This more accurately reflects macro.
0000    47 ;
0000    48 ; 15    Bill Landry     4 June 80
0000    49 ;       Deleted BUS/PRB.RD.PTE from macro CLRTB.VA_VA and CLRCH.VA_VA to
0000    50 ;       satisfy validity checks.
0000    51 ;
0000    52 ; 14    Dave Spain      23 May 80
0000    53 ;       Added BUS/PRB.RD.PTE to the clear cache and clear TB macros for Mr.
0000    54 ;       Bill.
0000    55 ;
0000    56 ; 13    Dave Spain      14 Apr 80
0000    57 ;       Added SIZE[] or VSIZE/1 to some macros. This allows version 65 of
0000    58 ;       the COMET micro-code defines to run.
0000    59 ;
```

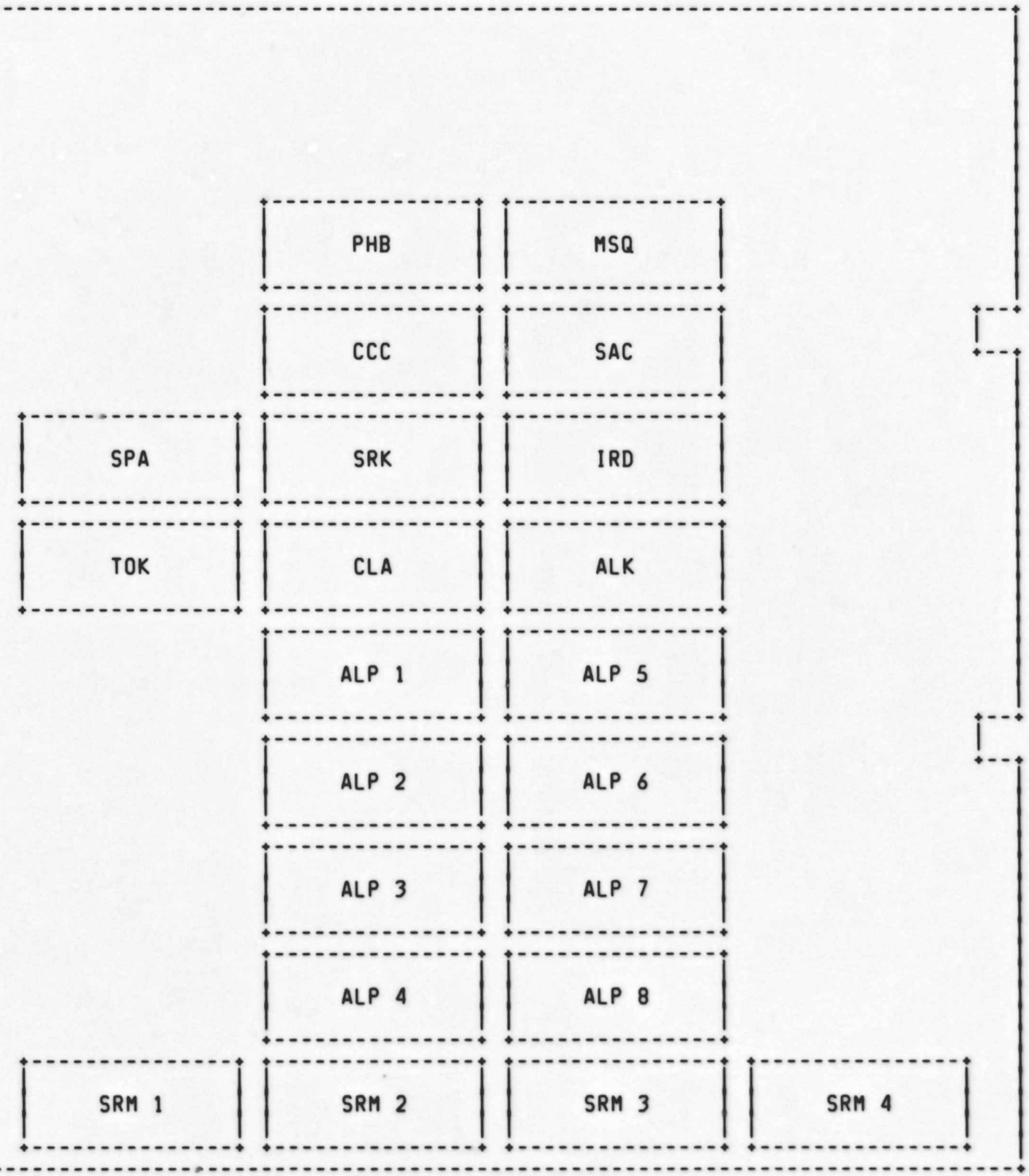
```
0000     60 ; 12    Bill Landry      1 Apr 80
0000     61 ;       Added BUS/PRB.RD micro order to RDM_TB macro to compensate for timing
0000     62 ;       problem in ADD gate array.  This is not an APRIL FOOL!
0000     63 ;
0000     64 ; 11    Dave Spain       27 Mar 80
0000     65 ;       Fixed TB macro's to include MSRC/VA to prevent validity failure.
0000     66 ;
0000     67 ; 10    Bill Landry      19 Mar 80
0000     68 ;       Added Q_R[].RL.P and VA_Q.OR.R[]
0000     69 ;
0000     70 ; 09    Bill Landry      19 Mar 80
0000     71 ;       Deleted TB_R[].RL.P.OR.D
0000     72 ;
0000     73 ; 08    Bill Landry      18 Mar 80
0000     74 ;       Added VA_R[].RL.P
0000     75 ;
0000     76 ; 07    Bill Landry      07 Mar 80
0000     77 ;       Added Q_M[].RL.PTE
0000     78 ;
0000     79 ; 06    Dave Spain       06 Mar 80
0000     80 ;       Changed TB_R[].RL.P macro to TB_R[].RL.P.OR.D.
0000     81 ;
0000     82 ; 05    Dave Spain       03 Mar 80
0000     83 ;       Added TB_R[].RL.P macro for Bill Landry.
0000     84 ;
0000     85 ; 04    Dave Spain       22 Feb 80
0000     86 ;       Added CLRTB.VA_R[] macro for Bill Landry.
0000     87 ;
0000     88 ; 03    Dave Spain       30 Jan 80
0000     89 ;       Added Tom Groetzinger's macro definitions.
0000     90 ;
0000     91 ; 02    Dave Spain       30 Jan 80
0000     92 ;       Added Tony Vezza's macro definitions.
0000     93 ;
0000     94 ; 01    Dave Spain       29 Jan 80
0000     95 ;       Macros initiated.
0000     96 ;--
0000     97
0000     98          .SBTTL  "       RDM Control File Macros"
0000     99
0000    100 ;STROBE.V.BUS            "RDMCF0/V.STROBE"
0000    101 ;DCS.ADDR_0              "RDMCF1/DCS.ADDR.CLR"
0000    102 ;STOP.CLOCK              "RDMCF2/STOP.CPU.CLOCK"
0000    103 ;LATCH.CS.ADDR           "RDMCF3/DIS.STROBE.CS.A"
0000    104 ;RDM_WB                  "RDMCF4/H.REG.FROM.WB"
0000    105 ;WB_RDM                  "RDMCF5/WB.FROM.D.REG"
0000    106 ;RDM_CMI                                "RDMCF6/STROBE.CMI"
0000    107 ;INH.CLOCK.SA            "RDMCF7/INH.SA.CLOCK"
0000    108
0000    109          .SBTTL  "       Diagnostic Macros"
0000    110
0000    111 ;CLOCK.EXT                              "CLKX/XTND"
0000    112 ;CLRCH.VA_RDM                           "WB_RDM,WCTRL/CLRCH.VA_WB,BUS/PRB.RD.PTE"
0000    113 ;CLRCH.VA_VA                            "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRCH.VA_WB
0000    114 ;CLRTB.VA_D                             "RSRC/ZERO,MUX/D.R1,ALU/OR,WCTRL/CLRTB.VA_WB,
0000    115 ;                                           BUS/PRB.RD.PTE"
0000    116 ;CLRTB.VA_RDM                           "WB_RDM,WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
```

```
0000   117 ;CLRTB.VA_R[]                    "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/CLRTB.VA_WB,
0000   118 ;                                        BUS/PRB.RD.PTE"
0000   119 ;CLRTB.VA_VA                     "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRTB.VA_WB
0000   120 ;CLRTB.VA_WB                     "WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
0000   121 ;D_LONLIT                       "RSRC/LONLIT,ALPCTL/WX_D_R.Q_D"
0000   122 ;D_M[]_LONLIT                   "D_LONLIT,MSRC/a1"
0000   123 ;D_VA_0                         "RSRC/ZERO,ROT/ZERO,MUX/R.S,ALU/OR,DQ1/D_WX,
0000   124 ;                                        WCTRL/VA_WB"
0000   125 ;FPA_M[] FPA_RDM                        "MSRC/a1,WB_RDM,FPA/FPA_MBUS.FPA_WBUS"
0000   126 ;FPA_M[] FPA_R[]                       "FPA/FPA_MBUS.FPA_WBUS,MSRC/a1,RSRC/a2,MUX/R
0000   127 ;                                        ROT/ZERO,ALU/OR"
0000   128 ;FPA_RDM                              "WB_RDM,FPA/FPA_DATA.WBUS"
0000   129 ;FPA_R[]                             "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,FPA/FPA_DAT
0000   130 ;FPA_R[]+M[]                        "RSRC/a1,MSRC/a2,MUX/M.R1,ALU/A+B+CI,ALUCI/ZERO,FPA/
0000   131 ;FPA_WB                            "FPA/FPA_DATA.WBUS"
0000   132 ;MEMSCAR_R[]                       "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/MEMSCAR_WB"
0000   133 ;M[]_FPA                           "MSRC/a1,SPW/MLONG,FPA/WBUS_FPA"
0000   134 ;M[]_LONLIT                       "MSRC/a1,SPW/MLONG,ALPCTL/WX_R.Q_D,RSRC/LONLIT"
0000   135 ;M[]_R[].OR.NIB0(M)               "MSRC/a1,RSRC/a2,SPW/MLONG,ROT/GETNIB,ALU/OR,
0000   136 ;                                        MUX/R.S,DQ1/NOP"
0000   137 ;M[]_RDM                           "WB_RDM,SPW/MLONG,MSRC/a1"
0000   138 ;PC_PC+RDM                        "WCTRL/PC_PC+WB,WB_RDM"
0000   139 ;PC_PC+4 MB_XB                    "MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,SIZE[LONG]"
0000   140 ;PC_PC+4 RDM_XB                   "RSRC/ZERO,MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,
0000   141 ;                                        MUX/M.R1,ALU/OR,RDM_WB,SIZE[LONG]"
0000   142 ;PC_RDM                           "WB_RDM,WCTRL/PC_WB"
0000   143 ;PSL_RDM                          "CCPSL/PSL_WB.CCBR_ALUS,WB_RDM"
0000   144 ;PSL_WB                           "CCPSL/PSL_WB.CCBR_ALUS"
0000   145 ;Q_D_WX_R                         "ALPCTL/WX_R.Q_D"
0000   146 ;Q_LONLIT                        "RSRC/LONLIT,ALPCTL/WX_S.Q_R"
0000   147 ;Q_M[].RL.PTE                     "ALPCTL/WX_Q_S,MSRC/a1,ROT/RL.MM.PTE"
0000   148 ;Q_R[].AND.Q                      "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/AND"
0000   149 ;Q_R[].ASL.P                      "ALPCTL/WX_Q_S,RSRC/a1,ROT/ASL.R.P"
0000   150 ;Q_R[].OR.Q                       "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/OR"
0000   151 ;Q_R[].RL.P                       "ALPCTL/WX_Q_S,RSRC/a1,ROT/RL.RR.P"
0000   152 ;RDM_ALUF                        "ALPCTL/WB_ALUF,RDM_WB"
0000   153 ;RDM_FPA                          "FPA/WBUS_FPA,RDM_WB"
0000   154 ;RDM_LONLIT                      "WB_LONLIT,RDM_WB"
0000   155 ;RDM_LONLIT.Q_M                   "RSRC/LONLIT,ALPCTL/WX_R.Q_M,RDM_WB"
0000   156 ;RDM_M[]                         "MSRC/a1,ROT/ZERO,MUX/M.S,ALU/OR,RDM_WB"
0000   157 ;RDM_M[].OR.R[]                  "WB_M[a1].OR.R[a2],RDM_WB"
0000   158 ;RDM_M[]-R[]                     "WB_M[a1]-R[a2],RDM_WB"
0000   159 ;RDM_M[].XZ.OR.R[]               "ALU/A+B+CI,MUX/R.S,ROT/XZ.MM,MSRC/a1,RSRC/a2,
0000   160 ;                                RDM_WB"
0000   161 ;RDM_PC                          "MSRC/PC,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
0000   162 ;RDM_PCBACK                      "RSRC/ZERO,MSRC/PCBACK,MUX/M.R1,ALU/OR,RDM_WB"
0000   163 ;RDM_Q                           "RSRC/ZERO,MUX/R.Q,ALU/OR,RDM_WB"
0000   164 ;RDM_Q Q_D                       "ALPCTL/WX_Q.Q_D,RDM_WB"
0000   165 ;RDM_R[]                         "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,RDM_WB"
0000   166 ;RDM_TB                          "WB_M[TB],RDM_WB,BUS/PRB.RD,SIZE[LONG]"
0000   167 ;RDM_VA                          "MSRC/VA,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
0000   168 ;RNUM_LONLIT                     "ALPCTL/WX_D_R.Q_M,MSRC/RNUM_WBUS,RSRC/LONLIT"
0000   169 ;R[]_RDM                         "WB_RDM,SPW/RLONG,RSRC/a1"
0000   170 ;R[].DT_Q                       "RSRC/a1,D_Q Q_D,SPW/RSIZE,VSIZE/1"
0000   171 ;R[]_VA                          "RSRC/a1,MSRC/VA,ROT/ZERO,MUX/M.S,ALU/OR,SPW/RLONG"
0000   172 ;R[]_WB                          "RSRC/a1,SPW/RLONG"
0000   173 ;TB_RDM                          "WCTRL/TB_WB,WB_RDM,MSRC/VA"
```

```
0000   174 ;TB_WB                              "WCTRL/TB_WB,MSRC/VA"
0000   175 ;TESTFPA []                         "TESTFPA/a1"
0000   176 ;VA_PC+LONLIT+I.PC_PC+I             "RSRC/LONLIT,MSRC/XB.PC_PC+I,ROT/ZERO,MUX/R.S,ALU/OR
0000   177 ;                                           ISTRM/ISIZE_DSIZE,WCTRL/VA_PC+I+W.PC_PC+I,
0000   178 ;                                           SIZE[LONG]"
0000   179 ;VA_Q.OR.R[]                         "WCTRL/VA_WB,RSRC/a1,ALU/OR,MUX/R.Q"
0000   180 ;VA_RDM                             "WB_RDM,WCTRL/VA_WB"
0000   181 ;VA_R[].RL.P                        "ALPCTL/WX_S,ROT/RL.RR.P,RSRC/a1,WCTRL/VA_WB"
0000   182 ;VA_VA-ZLIT0[]                      "WCTRL/VA_WB,LIT/LITRL,LITRL/a1,ROT/ZLIT0,MSRC/VA,
0000   183 ;                                           MUX/M.S,ALU/A-B-CI"
0000   184 ;VA_WB                              "WCTRL/VA_WB"
0000   185 ;WB_FPA CCR                         "FPA/WBUS_FPA.CC"
0000   186 ;WB_FPA TCR                         "WCTRL/FPTCR"
0000   187 ;WB_LONLIT                          "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
0000   188 ;WB_LONLIT.Q_M                      "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
0000   189 ;WB_M[]+R[]                         "ALU/A+B+CI,ALUCI/ZERO,MUX/M.R1,MSRC/a1,RSRC/a2"
0000   190 ;WB_R[].Q_D                        "RSRC/a1,ALPCTL/WX_R.Q_D"
0000   191 ;WDR_R[]                                 "WCTRL/WDR_WB,ALU/OR,MUX/R.S,RSRC/a1,ROT/ZER
0000   192 ;WRITE.LONG R[]                    "BUS/WRITE.LNG,WCTRL/WDR_WB.UR,RSRC/a1,ROT/ZERO,
0000   193 ;                                           MUX/R.S,ALU/OR,SIZE[LONG]"
0000   194         .SBTTL  ""
0000   195         .SBTTL  "MODULE GATE ARRAY MAPS"
```

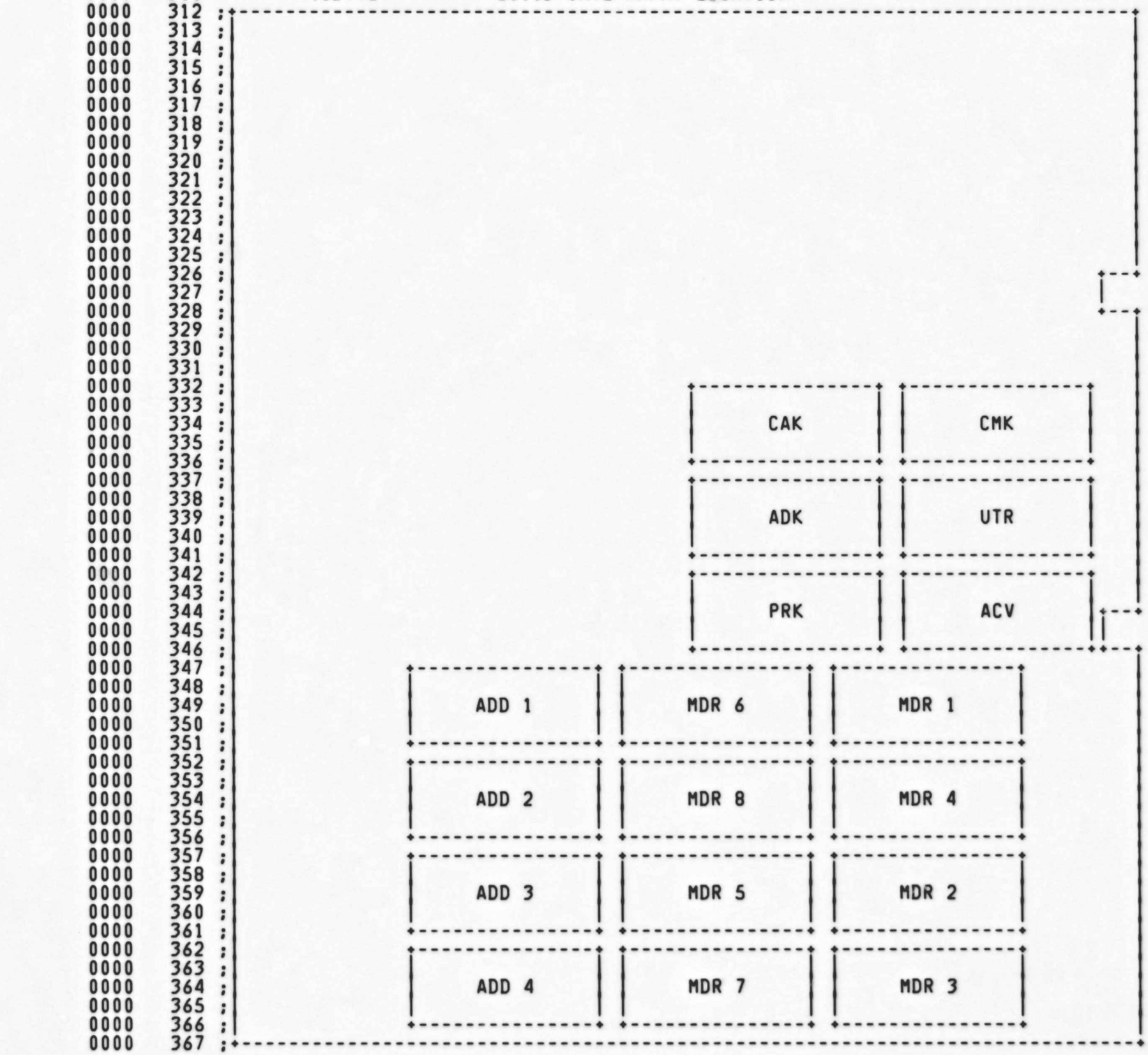
```
                        197          .SBTTL  "       L0001 MULTIPLIER AND GATE ARRAY LOCATION"
```





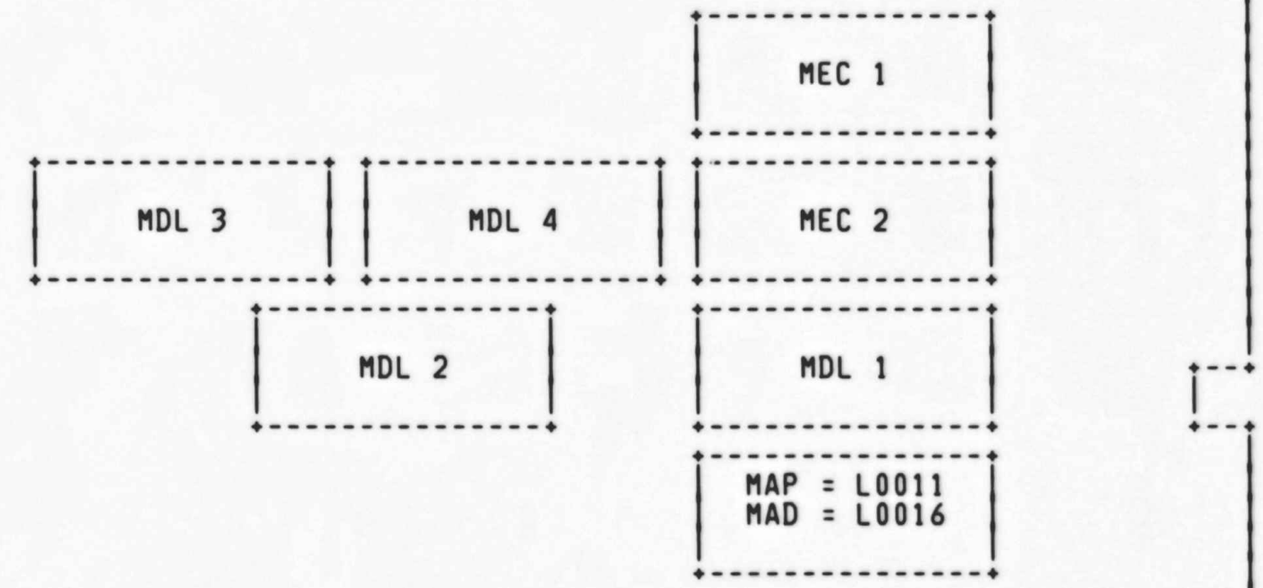
```
0000          .SBTTL  "      L0002 GATE ARRAY LOCATION"
0000   ;+-----------------------------------------------------------------------------+
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                    +---------------+ +---------------+                      |
0000   ;|                    |               | |               |                      |
0000   ;|                    |      PHB      | |      MSQ      |                      |
0000   ;|                    |               | |               |                      |
0000   ;|                    +---------------+ +---------------+                      |
0000   ;|                    +---------------+ +---------------+                  +---+
0000   ;|                    |               | |               |                  |   |
0000   ;|                    |      CCC      | |      SAC      |                  +---+
0000   ;|                    |               | |               |                      |
0000   ;|                    +---------------+ +---------------+                      |
0000   ;|   +---------------+ +---------------+ +---------------+                      |
0000   ;|   |               | |               | |               |                      |
0000   ;|   |      SPA      | |      SRK      | |      IRD      |                      |
0000   ;|   |               | |               | |               |                      |
0000   ;|   +---------------+ +---------------+ +---------------+                      |
0000   ;|   +---------------+ +---------------+ +---------------+                      |
0000   ;|   |               | |               | |               |                      |
0000   ;|   |      TOK      | |      CLA      | |      ALK      |                      |
0000   ;|   |               | |               | |               |                      |
0000   ;|   +---------------+ +---------------+ +---------------+                      |
0000   ;|                    +---------------+ +---------------+                      |
0000   ;|                    |               | |               |                      |
0000   ;|                    |      ALP 1    | |      ALP 5    |                      |
0000   ;|                    |               | |               |                      |
0000   ;|                    +---------------+ +---------------+                  +---+
0000   ;|                    +---------------+ +---------------+                  |   |
0000   ;|                    |               | |               |                  +---+
0000   ;|                    |      ALP 2    | |      ALP 6    |                      |
0000   ;|                    |               | |               |                      |
0000   ;|                    +---------------+ +---------------+                      |
0000   ;|                    +---------------+ +---------------+                      |
0000   ;|                    |               | |               |                      |
0000   ;|                    |      ALP 3    | |      ALP 7    |                      |
0000   ;|                    |               | |               |                      |
0000   ;|                    +---------------+ +---------------+                      |
0000   ;|                    +---------------+ +---------------+                      |
0000   ;|                    |               | |               |                      |
0000   ;|                    |      ALP 4    | |      ALP 8    |                      |
0000   ;|                    |               | |               |                      |
0000   ;|                    +---------------+ +---------------+                      |
0000   ;|   +---------------+ +---------------+ +---------------+ +---------------+    |
0000   ;|   |               | |               | |               | |               |    |
0000   ;|   |     SRM 1     | |     SRM 2     | |     SRM 3     | |     SRM 4     |    |
0000   ;|   |               | |               | |               | |               |    |
0000   ;|   +---------------+ +---------------+ +---------------+ +---------------+    |
0000   ;+-----------------------------------------------------------------------------+
```

```
                        .SBTTL  "       L0003 GATE ARRAY LOCATION"
```



```
                        0000   369          .SBTTL  "       L0004 GATE ARRAY LOCATION"
                        0000   370 ;+-------------------------------------------------------------------------------+
                        0000   371 ;|                                                                               |
                        0000   372 ;|                                                                               |
                        0000   373 ;|                                                                               |
                        0000   374 ;|                                                                               |
                        0000   375 ;|                                                                               |
                        0000   376 ;|                                                                               |
                        0000   377 ;|                                                                               |
                        0000   378 ;|                                                                               |
                        0000   379 ;|                                                                               |
                        0000   380 ;|                                                                               |
                        0000   381 ;|                          +---------------+                                    |
                        0000   382 ;|                          |               |                                    |
                        0000   383 ;|                          |      UCN      |                                    |
                        0000   384 ;|                          |               |                                    |
                        0000   385 ;|                          +---------------+                                    |
                        0000   386 ;|                          +---------------+                                    |
                        0000   387 ;|                          |               |                                    |
                        0000   388 ;|                          |     UDP 1     |                                    |
                        0000   389 ;|                          |               |                                    |
                        0000   390 ;|                          +---------------+                                    |
                        0000   391 ;|                          +---------------+                                    |
                        0000   392 ;|                          |               |                                    |
                        0000   393 ;|                          |     UDP 2     |                                    |
                        0000   394 ;|                          |               |                                    |
                        0000   395 ;|                          +---------------+                                    |
                        0000   396 ;|                          +---------------+                                    |
                        0000   397 ;|                          |               |                                    |
                        0000   398 ;|                          |     UDP 4     |                                    |
                        0000   399 ;|                          |               |                                    |
                        0000   400 ;|                          +---------------+                                    |
                        0000   401 ;|                          +---------------+                                    |
                        0000   402 ;|                          |               |                                    |
                        0000   403 ;|                          |     UDP 3     |                                    |
                        0000   404 ;|                          |               |                                    |
                        0000   405 ;|                          +---------------+                                    |
                        0000   406 ;|                          +---------------+                                    |
                        0000   407 ;|                          |               |                                    |
                        0000   408 ;|                          |      INT      |                                    |
                        0000   409 ;|                          |               |                                    |
                        0000   410 ;|                          +---------------+                                    |
                        0000   411 ;|                          +---------------+                                    |
                        0000   412 ;|                          |               |                                    |
                        0000   413 ;|                          |     CON 1     |                                    |
                        0000   414 ;|                          |               |                                    |
                        0000   415 ;|                          +---------------+                                    |
                        0000   416 ;|                          +---------------+                                    |
                        0000   417 ;|                          |               |                                    |
                        0000   418 ;|                          |     CON 2     |                                    |
                        0000   419 ;|                          |               |                                    |
                        0000   420 ;|                          +---------------+                                    |
                        0000   421 ;|                                                                               |
                        0000   422 ;|                                                                               |
                        0000   423 ;|                                                                               |
                        0000   424 ;|                                                                               |
                        0000   425 ;+-------------------------------------------------------------------------------+
```

```
                        427          .SBTTL  "       L0011 & L0016 GATE ARRAY LOCATION"
                        428 ;+-----------------------------------------------------------------------------+
                        429 ;|                                                                             |
                        430 ;|                                                                             |
                        431 ;|                                                                             |
                        432 ;|                                                                             |
                        433 ;|                                                                             |
                        434 ;|                                                                             |
                        435 ;|                                                                             |
                        436 ;|                                                                             |
                        437 ;|                                                                             |
                        438 ;|                                                                             |
                        439 ;|                                                                             |
                        440 ;|                                                                             |
                        441 ;|                                                                             |
                        442 ;|                                                                             |
                        443 ;|                                                                             |
                        444 ;|                                                                             |
                        445 ;|                                                                             |
                        446 ;|                                                                             |
                        447 ;|                                                                             |
                        448 ;|                                                                             |
                        449 ;|                                                                             |
                        450 ;|                                                                             |
                        451 ;|                                                                             |
                        452 ;|                                                                             |
                        453 ;|                                                                             |
                        454 ;|                                                                             |
                        455 ;|                                                                             |
                        456 ;|                                                                             |
                        457 ;|                                                                             |
                        458 ;|                                                                             |
                        459 ;|                                                                             |
                        460 ;|                                                                             |
                        461 ;|                                                                             |
                        462 ;|                                                                             |
                        463 ;|                                                                             |
                        464 ;|                                                                             |
                        465 ;|                                                                             |
                        466 ;|                                                                             |
                        467 ;|                                                                             |
                        468 ;|                                                                             |
                        469 ;|                                                                             |
                        470 ;|                                                                             |
                        471 ;|                                                                             |
                        472 ;|                                                                             |
                        473 ;|                                                                             |
                        474 ;|                                                                             |
                        475 ;|                                                                             |
                        476 ;|                                                                             |
                        477 ;|                                                                             |
                        478 ;|                                                                             |
                        479 ;|                                                                             |
                        480 ;|                                                                             |
                        481 ;|                                                                             |
                        482 ;|                                                                             |
                        483 ;+-----------------------------------------------------------------------------+
```



```
                        0000    485          .SBTTL  ""
                        0000    486          .LIST   MC
                        0000    487          .LIBRARY        /UCOD$0:[VAX750]COMETMAC/
                        0000    488          .LIST   ME
                        0000    489          .NLIST  CND,MC
                        0000                 .SBTTL  "EQUATED SYMBOLS
                        0000         ;
                        0000         ; RDM REGISTER DEFINITIONS:
                        0000         ;
                        0000         ; NOTE: THESE REGITER DEFINITIONS ARE OFFSET BY THE LOAD/START ADDRESS OF
                        0000         ;       THE PROGRAM. FOR THE SIMULATOR VERSION THIS IS 400(X) AND FOR THE
                        0000         ;       REAL VERSION IT IS 8400(X).
                        0000         ;
00007400                0000                 DCS_DATA_REG     =       ^XF800 + HEAD
00007401                0000                 DCS_ADDR_REG_WO  =       ^XF801 + HEAD
00007402                0000                 DCS_CTRL_REG     =       ^XF802 + HEAD
00007403                0000                 DCS_CTRL_FILE    =       ^XF803 + HEAD
00007404                0000                 ADDR_MATCH_LO    =       ^XF804 + HEAD
00007405                0000                 ADDR_MATCH_HI    =       ^XF805 + HEAD
00007406                0000                 STATUS_REG       =       ^XF806 + HEAD
                        0000         ;
00007410                0000                 A_REG_BYTE_0     =       ^XF810 + HEAD
00007411                0000                 A_REG_BYTE_1     =       ^XF811 + HEAD
00007412                0000                 A_REG_BYTE_2     =       ^XF812 + HEAD
00007413                0000                 A_REG_BYTE_3     =       ^XF813 + HEAD
00007414                0000                 MD_REG_BYTE_0    =       ^XF814 + HEAD
00007415                0000                 MD_REG_BYTE_1    =       ^XF815 + HEAD
00007416                0000                 MD_REG_BYTE_2    =       ^XF816 + HEAD
00007417                0000                 MD_REG_BYTE_3    =       ^XF817 + HEAD
00007418                0000                 MH_REG_BYTE_0    =       ^XF818 + HEAD
00007419                0000                 MH_REG_BYTE_1    =       ^XF819 + HEAD
0000741A                0000                 MH_REG_BYTE_2    =       ^XF81A + HEAD
0000741B                0000                 MH_REG_BYTE_3    =       ^XF81B + HEAD
0000741C                0000                 CMI_CTRL_REG     =       ^XF81C + HEAD
0000741D                0000                 MAINT_REG        =       ^XF81D + HEAD
0000741E                0000                 RD_CTRL_REG      =       ^XF81E + HEAD
                        0000         ;
00007420                0000                 FRONT_PNL_1      =       ^XF820 + HEAD
00007421                0000                 FRONT_PNL_2      =       ^XF821 + HEAD
00007422                0000                 CS_ADD_TRAP_LOW  =       ^XF822 + HEAD
00007423                0000                 CS_ADD_TRAP_HGH  =       ^XF823 + HEAD
00007424                0000                 CS_ADD_BUFF_LOW  =       ^XF824 + HEAD
00007425                0000                 CS_ADD_BUFF_HGH  =       ^XF825 + HEAD
00007426                0000                 BUFF_ADDR_LOW    =       ^XF826 + HEAD
00007427                0000                 DCS_ADDR_REG_RO  =       ^XF827 + HEAD
                        0000         ;
0000742C                0000                 WD_REG_BYTE_0    =       ^XF82C + HEAD
0000742D                0000                 WD_REG_BYTE_1    =       ^XF82D + HEAD
0000742E                0000                 WD_REG_BYTE_2    =       ^XF82E + HEAD
0000742F                0000                 WD_REG_BYTE_3    =       ^XF82F + HEAD
00007430                0000                 WH_REG_BYTE_0    =       ^XF830 + HEAD
00007431                0000                 WH_REG_BYTE_1    =       ^XF831 + HEAD
00007432                0000                 WH_REG_BYTE_2    =       ^XF832 + HEAD
00007433                0000                 WH_REG_BYTE_3    =       ^XF833 + HEAD
                        0000         ;
                        0000         ; RDM REGISTER NAMES AS DEFINED IN HARDWARE SPEC
                        0000         ;
```

```
00007400  0000              DCSDA=DCS_DATA_REG              ;Diagnostic control store data register
00007401  0000              DCSAD=DCS_ADDR_REG_WO           ;Diagnostic control store address reg
00007402  0000              DCSCR=DCS_CTRL_REG              ;Diagnostic control store control reg
00007403  0000              DCSCF=DCS_CTRL_FILE             ;Diagnostic control store control file
00007404  0000              CSAMR=ADDR_MATCH_LO             ;Control store address match register
00007406  0000              STATUS=STATUS_REG               ;Status register
00007410  0000              MAREG=A_REG_BYTE_0              ;Memory address register
00007414  0000              MDREG=MD_REG_BYTE_0             ;Memory data register
00007418  0000              MHREG=MH_REG_BYTE_0             ;Memory holding register
0000741C  0000              CMICTL=CMI_CTRL_REG             ;CMI control
0000741D  0000              MAIN32=MAINT_REG                ;32 Bit path maintenance control
0000741E  0000              RDCTRL=RD_CTRL_REG              ;Remote diagnosis control
00007420  0000              FRONT1=FRONT_PNL_1              ;Front panel readback 1
00007421  0000              FRONT2=FRONT_PNL_2              ;Front panel readback 2
00007422  0000              CSTRP=CS_ADD_TRAP_LOW           ;Control store address trap register
00007424  0000              CSBUF=CS_ADD_BUFF_LOW           ;Control store address buffer
00007426  0000              BADD1=BUFF_ADDR_LOW             ;Control store address trace readback
00007427  0000              CLKDCS=DCS_ADDR_REG_RO          ;Clock control & DCS address register
0000742C  0000              WDREG=WD_REG_BYTE_0             ;WBUS data register
00007430  0000              WHREG=WH_REG_BYTE_0             ;WBUS holding register
          0000        ;
          0000        ; RDM REGISTER BIT DEFINITIONS:
          0000        ;
          0000        ; DCS CONTROL REGISTER
00000001  0000              HALT_ON_MATCH    =      1
00000002  0000              CLEAR_CTRL_FILE  =      2
00000004  0000              TRACE_ENABL      =      4
00000008  0000              PAR_CHK_ENABL    =      8
00000010  0000              PAR_STOP_ENABL   =      ^X10
00000020  0000              CLK_CTRL_0       =      ^X20    ; ACTIVE LOW
00000040  0000              CLK_CTRL_1       =      ^X40    ; ACTIVE LOW
00000080  0000              MICRO_ADDR_INH   =      ^X80
          0000            ;
          0000            ; FOLLOWING IS A FUNCTION TABLE OF THE CLOCK CONTROL BITS
          0000            ;
          0000                ;       CLK_CTRL 1,0    FUNCTION
          0000                ;           11            HALT
          0000                ;           01            SINGLE MICRO INSTRUCTION
          0000                ;           10            SINGLE TICK
          0000                ;           00            RUN
          0000        ;
          0000        ; DCS CONTROL FILE      ALL OF THESE BITS ARE ACTIVE LOW
          0000        ;
00000001  0000              V_BUS_STROBE     =      1
00000002  0000              DCS_ADDR_CLEAR   =      2
00000004  0000              STOP_CLOCK       =      4
00000008  0000              EN_TRACE_REG     =      8
00000010  0000              H_FROM_WBUS      =      ^X10
00000020  0000              WBUS_FROM_D      =      ^X20
00000040  0000              STROBE_CMI       =      ^X40
00000080  0000              SA_CLOCK         =      ^X80
          0000        ;
          0000        ; CS ADDRESS MATCH REGISTER (HIGH)
          0000        ;
00000040  0000              VBUS_SERIAL_IN   =      ^X40    ; ACTIVE LOW
00000080  0000              VBUS_CLOCK       =      ^X80
          0000        ;
```

```
                        0000    ; STATUS REGISTER
                        0000    ;
00000001                0000            VBUS_SERIAL_OUT =       1
00000002                0000            TRAP_OFF        =       2
00000008                0000            CMI_STATUS_0    =       8
00000010                0000            CMI_STATUS_1    =       ^X10
                        0000            ;
                        0000            ; THE CMI STATUS BITS ARE ENCODED AS FOLLOWS:
                        0000            ;
                        0000            ;       00 = NO RESPONSE
                        0000            ;       01 = UNCORRECTABLE ERROR
                        0000            ;       10 = CORRECTABLE ERROR
                        0000            ;       11 = NO ERRORS
00000020                0000            CMI_COMPLETE    =       ^X20
                        0000    ;
00000080                0000            CLOCK_STOPED    =       ^X80
                        0000    ;
                        0000    ; CMI CONTROL REGISTER
                        0000    ;
00000001                0000            CMI_CTRL_CLEAR  =       1
00000008                0000            CMI_GO          =       8
00000020                0000            CMI_WRITE       =       ^X20
00000040                0000            CMI_READ        =       ^X40
00000080                0000            SA_START_STOP   =       ^X80
                        0000    ;
                        0000    ; 32 BIT MAINTENANCE CONTROL REGISTER
                        0000    ;
00000001                0000            MAINT_TRAP_OFF  =       1
00000002                0000            MAINT_STB_INH   =       2
00000004                0000            MAINT_DCS_ENABL =       4
00000008                0000            MAINT_WB_TO_WH  =       8
00000010                0000            MAINT_WD_TO_WB  =       ^X10
00000020                0000            MAINT_CMI_TO_MH =       ^X20
00000040                0000            MAINT_MD_TO_CMI =       ^X40
00000080                0000            MAINT_A_TO_CMI  =       ^X80
                        0000    ;
                        0000    ; RD CONTROL REGISTER
                        0000    ;
00000008                0000            MASTER_HALT_EN  =       8
00000010                0000            VBUS_LOAD       =       ^X10
00000020                0000            TRAP_HALT_EN    =       ^X20
00000040                0000            DC_LOW          =       ^X40
00000080                0000            AC_LOW          =       ^X80
                        0000    ;
                        0000    ; FRONT PANEL REGISTER 1
                        0000    ;
00000001                0000            FP_BOOT_0       =       1
00000002                0000            FP_BOOT_1       =       2
00000004                0000            FP_START_0      =       4
00000008                0000            FP_START_1      =       8
00000010                0000            CPU_RUN         =       ^X10
00000020                0000            PARITY_ERROR    =       ^X20
                        0000    ;
                        0000    ; FRONT PANEL REGISTER 2
                        0000    ;
00000001                0000            REMOTE          =       1
00000002                0000            FAULT           =       2
```

```
00000004  0000                TEST            =       4
          0000        ;
          0000        ;
          0000        ;
00000040  0000                PB_INIT         =       ^X40    ; ACTIVE LOW
00000080  0000                FRONT_PNL_LOCK  =       ^X80
          0000        ;
          0000        ; DCS ADDRESS REGISTER READ ONLY
          0000        ;
00000040  0000                CLK_CTRL_0_RO   =       ^X40    ; ACTIVE HIGH
00000080  0000                CLK_CTRL_1_RO   =       ^X80    ; ACTIVE HIGH
          0000        ;
          0000        ; MISCELLANEOUS EQUATES:
          0000        ;
00000001  0000                BYTE            =       1
00000002  0000                WORD            =       2
00000004  0000                LONG            =       4
0000000A  0000                MICROWORD       =       10
00002C5E  0000                MONITOR_SIZE    =       ^X2C5E
00000400  0000                STACK_SIZE      =       1024
000007A2  0000                M$TEST_SIZE     =       14336 - STACK_SIZE - MONITOR_SIZE
          0000                                              ; MAXIMUM SIZE OF A TEST OVERLAY
000003E2  0000                M$TEST_SIZE_D   = M$TEST_SIZE - 960
          0000                                              ; MAXIMUM SIZE OF TEST OVERLAY WITH
          0000                                              ; 'DEBUG' ENABLED IN
00000001  0000                B               =       BYTE
00000002  0000                W               =       WORD
00000004  0000                L               =       LONG
0000000A  0000                M               =       MICROWORD
00000001  0000                HIGH            =       1
00000000  0000                LOW             =       0
00000000  0000                ONERROR         =       0
00000001  0000                NOERROR         =       1
00000002  0000                ONEQUAL         =       2
00000003  0000                NOTEQUAL        =       3
00000010  0000                MAX_ARGUMENTS   =       16      ; BYTE LENGTH OF LARGEST PSEUDO OP
00000036  0000                DCS_SCRATCH_ADR =       54      ; DCS SCRATCH ADDRESS START
000032FC  0000                ERROR_LOG_ADR   =       ^XB6FC + HEAD
          0000                                              ; START ADDRESS OF MEMORY ERROR LOG
          0000    491         .NLIST  ME
          0000    492         .LIST   MC
          0000    493         .SBTTL  ""
          0000    494         .NLIST  MC      ; SHUT-OFF LISTING FOR FIRST 'NEWTEST' CALL
```

```
0008     2 .END
```

```
$CONTROL_C_CODE= 0000000A
$FILE_NOT_FOUND= 00000009
ACV             = 0000000F
AC_LOW          = 00000080
ADD             = 00000019
ADDR_MATCH_HI   = 00007405
ADDR_MATCH_LO   = 00007404
ADK             = 0000000B
ALK             = 00000000
ALP             = 00000016
A_REG_BYTE_0    = 00007410
A_REG_BYTE_1    = 00007411
A_REG_BYTE_2    = 00007412
A_REG_BYTE_3    = 00007413
B               = 00000001
BADD1           = 00007426
BELL_FLAG       = 00000008
BELL_KEY        = 0000000E
BUFF_ADDR_LOW   = 00007426
BURST_STOP_FLAG= 00000080
BYTE            = 00000001
CAK             = 0000000A
CCC             = 00000004
CCS             = 00000005
CF_KEY          = 00000024
CLA             = 00000003
CLEAR_CTRL_FILE= 00000002
CLKDCS          = 00007427
CLK_CTRL_0      = 00000020
CLK_CTRL_0_RO   = 00000040
CLK_CTRL_1      = 00000040
CLK_CTRL_1_RO   = 00000080
CLOCK_STOPED    = 00000080
CMI             = 00000007
CMICTL          = 0000741C
CMI_COMPLETE    = 00000020
CMI_CTRL_CLEAR  = 00000001
CMI_CTRL_REG    = 0000741C
CMI_GO          = 00000008
CMI_READ        = 00000040
CMI_STATUS_0    = 00000008
CMI_STATUS_1    = 00000010
CMI_WRITE       = 00000020
CML             = 0000000E
CON             = 00000011
CONTINUE_FLAG   = 00000002
CONTROL_C_FLAG  = 00000001
CONT_FLAG       = 00000020
CONT_KEY        = 0000001C
CPU_RUN         = 00000010
CSAMR           = 00007404
CSBUF           = 00007424
CSTRP           = 00007422
CS_ADD_BUFF_HGH= 00007425
CS_ADD_BUFF_LOW= 00007424
CS_ADD_TRAP_HGH= 00007423
CS_ADD_TRAP_LOW= 00007422
CYCLE_FLAG      = 00000008
CYCLE_KEY       = 00000020
DCSAD           = 00007401
DCSCF           = 00007403
DCSCR           = 00007402
DCSDA           = 00007400
DCS_ADDR_CLEAR  = 00000002
DCS_ADDR_REG_RO= 00007427
DCS_ADDR_REG_WO= 00007401
DCS_CTRL_FILE   = 00007403
DCS_CTRL_REG    = 00007402
DCS_DATA_REG    = 00007400
DCS_SCRATCH_ADR= 00000036
DCS_START       = 00001800
DC_LOW          = 00000040
DIAGNOSE_FLAG   = 00000004
DPM             = 00000000
END_SLICE       = 00000025
EN_TRACE_REG    = 00000008
ERROR_FLAG      = 00000010
ERROR_LOG_ADR   = 000032FC
ERR_LOG_INIT    = 00000001
EXP_STALL_FLAG  = 00000010
EXP_UTRAP_FLAG  = 00000040
FAC             = 00000020
FAULT           = 00000002
FCC             = 00000015
FCS             = 00000021
FETCH_FLAG      = 00000008
FEX             = 0000001E
FFH             = 00000024
FFL             = 00000023
FIC             = 0000001F
FIO             = 0000001D
FLAG_KEY        = 00000004
FMR             = 00000022
FPA             = 00000002
FP_BOOT_0       = 00000001
FP_BOOT_1       = 00000002
FP_START_0      = 00000004
FP_START_1      = 00000008
FQA             = 00000014
FRONT1          = 00007420
FRONT2          = 00007421
FRONT_PNL_1     = 00007420
FRONT_PNL_2     = 00007421
FRONT_PNL_LOCK  = 00000080
HALT_FLAG       = 00000001
HALT_KEY        = 00000008
HALT_ON_MATCH   = 00000001
HEAD            = FFFF7C00
HIGH            = 00000001
H_FROM_WBUS     = 00000010
IB_FLAG         = 00000020
IB_KEY          = 00000012
INSTR_FLAG      = 00000004
INSTR_KEY       = 00000018
INT             = 00000010
IRD             = 00000005
L               = 00000004
LONG            = 00000004
LOOP_ERROR_FLAG= 00000020
LOOP_FLAG       = 00000002
LOOP_KEY        = 0000000A
LOST_FLAG       = 00000010
LOST_KEY        = 0000001A
LOW             = 00000000
M               = 0000000A
M$TEST_SIZE     = 000007A2
M$TEST_SIZE_D   = 000003E2
MA0             = 00000008
MA1             = 0000000A
MA2             = 0000000C
MAD             = 00000013
MAIN32          = 0000741D
MAINT_A_TO_CMI  = 00000080
MAINT_CMI_TO_MH= 00000020
MAINT_DCS_ENABL= 00000004
MAINT_MD_TO_CMI= 00000040
MAINT_REG       = 0000741D
MAINT_STB_INH   = 00000002
MAINT_TRAP_OFF  = 00000001
MAINT_WB_TO_WH  = 00000008
MAINT_WD_TO_WB  = 00000010
MAP             = 00000012
MAREG           = 00007410
MASTER_HALT_EN  = 00000008
MAX_ARGUMENTS   = 00000010
MA_KEY          = 00000038
MC0             = 00000003
MC1             = 00000009
MC2             = 0000000B
MDL             = 0000001B
MDR             = 00000018
MDREG           = 00007414
MD_KEY          = 00000034
MD_REG_BYTE_0   = 00007414
MD_REG_BYTE_1   = 00007415
MD_REG_BYTE_2   = 00007416
MD_REG_BYTE_3   = 00007417
MEC             = 0000001A
MHREG           = 00007418
MH_REG_BYTE_0   = 00007418
MH_REG_BYTE_1   = 00007419
MH_REG_BYTE_2   = 0000741A
MH_REG_BYTE_3   = 0000741B
MIC             = 00000001
MICROWORD       = 0000000A
MICRO_ADDR_INH  = 00000080
MONITOR_SIZE    = 00002C5E
MSQ             = 00000008
MT_KEY          = 0000002E
NER_FLAG        = 00000004
NER_KEY         = 0000000C
```

NOERROR	= 00000001	SA_FLAG	= 00000010
NOTEQUAL	= 00000003	SA_KEY	= 00000010
ONEQUAL	= 00000002	SA_START_STOP	= 00000080
ONERROR	= 00000000	SBE_FLAG	= 00000020
OP_BEGIN_LOOP	= 00000008	SBE_KEY	= 00000042
OP_BURST_CLOCK	= 00000002	SF_KEY	= 00000040
OP_COMPARE_REG	= 00000004	SOMM_FLAG	= 00000001
OP_COMPARE_REGM	= 00000006	SOMM_KEY	= 00000006
OP_COMPARE_VB	= 00000020	SPA	= 00000002
OP_DUMPLOG	= 00000030	SRK	= 00000001
OP_ENABL_STALL	= 00000038	SRM	= 00000017
OP_ENABL_TRAP	= 0000002E	SR_KEY	= 0000003C
OP_END_FILE	= 00000024	STACK_SIZE	= 00000400
OP_END_LOOP	= 0000000A	STATUS	= 00007406
OP_END_TEST	= 0000000C	STATUS_REG	= 00007406
OP_ERRLOG	= 00000032	STEP_KEY	= 0000001E
OP_ERROR_LOOP	= 0000000E	STOP_CLOCK	= 00000004
OP_FETCH	= 00000022	STROBE_CMI	= 00000040
OP_IF_ERROR	= 00000012	ST_KEY	= 0000001E
OP_INC_INDEX	= 0000002A	TEMP	= 00000002
OP_INIT	= 00000014	TEMP1	= 00000002
OP_LOAD_DCS	= 00000000	TEST	= 00000004
OP_LOAD_FIELD	= 00000026	TEST_KEY	= 00000000
OP_LOAD_INDEX	= 00000028	TICK_FLAG	= 00000002
OP_LOAD_REG	= 00000016	TICK_KEY	= 00000022
OP_MASK	= 00000018	TOK	= 00000006
OP_NEW_TEST	= 0000001A	TRACE_ENABL	= 00000004
OP_PATTERN_GEN	= 00000010	TRAP_HALT_EN	= 00000020
OP_PRINT_N	= 00000036	TRAP_OFF	= 00000002
OP_PRINT_S	= 00000034	TR_FLAG	= 00000080
OP_SAVE_INDEX	= 0000002C	TR_KEY	= 00000016
OP_SKIP	= 0000001C	TSTSPAN_FLAG	= 00000040
OP_SUB_TEST	= 0000001E	UBI	= 00000006
PARITY_ERROR	= 00000020	UDP	= 0000001C
PAR_CHK_ENABL	= 00000008	UTR	= 0000000D
PAR_STOP_ENABL	= 00000010	VA_KEY	= 00000032
PASS_KEY	= 00000002	VBUS_CLOCK	= 00000080
PB_INIT	= 00000040	VBUS_COMPARE	= 00000080
PB_KEY	= 00000036	VBUS_LOAD	= 00000010
PC_KEY	= 00000030	VBUS_SERIAL_IN	= 00000040
PHB	= 00000007	VBUS_SERIAL_OUT	= 00000001
PRK	= 0000000C	VB_KEY	= 00000026
PS_KEY	= 0000003E	V_BUS_STROBE	= 00000001
QA_FLAG	= 00000040	W	= 00000002
QA_KEY	= 00000014	WBUS_FROM_D	= 00000020
QV_FLAG	= 00000002	WDREG	= 0000742C
QV_KEY	= 00000028	WD_KEY	= 0000003A
RDCTRL	= 0000741E	WD_REG_BYTE_0	= 0000742C
RDM	= 00000004	WD_REG_BYTE_1	= 0000742D
RDM_FLAG	= 00000004	WD_REG_BYTE_2	= 0000742E
RDM_KEY	= 0000002A	WD_REG_BYTE_3	= 0000742F
RDM_LAST	= 00000008	WHREG	= 00007430
RD_CTRL_REG	= 0000741E	WH_REG_BYTE_0	= 00007430
REMOTE	= 00000001	WH_REG_BYTE_1	= 00007431
RT_KEY	= 0000002C	WH_REG_BYTE_2	= 00007432
SAC	= 00000009	WH_REG_BYTE_3	= 00007433
SA_CLOCK	= 00000080	WORD	= 00000002

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
DIRECTORY	00000008 (8.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	23	00:00:00.12	00:00:01.85
Command processing	863	00:00:00.75	00:00:01.41
Pass 1	213	00:00:02.05	00:00:03.21
Symbol table sort	0	00:00:00.25	00:00:00.27
Pass 2	32	00:00:01.11	00:00:01.92
Symbol table output	0	00:00:00.12	00:00:00.22
Psect synopsis output	0	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1131	00:00:04.41	00:00:08.89

The working set limit was 2048 pages.
28279 bytes (56 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 285 non-local and 0 local symbols.
747 source lines were read in Pass 1, producing 10 object records in Pass 2.
23 pages of virtual memory were used to define 2 macros.

Macro library statistics

Macro library name	Macros defined
DISK$UCODPACK00:[VAX750]COMETMAC.MLB;1	2
SYS$COMMON:[SYSLIB]STARLET.MLB;2	0
TOTALS (all libraries)	2

552 GETS were required to define 2 macros.

There were no errors, warnings or information messages.

```
MACRO UCOD$0:[VAX750.DPM]TITLE+DISK$UCODPACK00:[VAX750]COMETASM+UCOD$0:[VAX750.DPM]ECKAB0/LIS=ECKAB0/OBJ=ECKAB0
```

Table of contents

(1) 24 "DPM Revision History"
(1) 1 ""
(1) 3 "DIAGNOSTIC MICROCODE MACROS"
(1) 98 " RDM Control File Macros"
(1) 109 " Diagnostic Macros"
(1) 194 ""
(1) 195 "MODULE GATE ARRAY MAPS"
(1) 197 " L0001 MULTIPLIER AND GATE ARRAY LOCATION"
(1) 254 " L0002 GATE ARRAY LOCATION"
(1) 311 " L0003 GATE ARRAY LOCATION"
(1) 369 " L0004 GATE ARRAY LOCATION"
(1) 427 " L0011 & L0016 GATE ARRAY LOCATION"
(1) 485 ""
(1) 490 "EQUATED SYMBOLS
(1) 493 ""
(1) 2 "TEST 001 Check for Patch Control Store (PCS)/Initialize PCS
(1) 391 "TEST 002 Test Control Store Address Bus Integrity
(1) 469 "TEST 003 Wbus Data Integrity
(1) 547 "TEST 004 Test CPU Clock
(1) 630 "TEST 005 Test CPU Clock Single Tick and Step
(1) 738 "TEST 006 Control Store Bus Integrity
(1) 1224 "TEST 007 Long Literal Register to WBUS Data Path
(1) 1316 "TEST 008 Long Literal Register Data Integrity
(1) 1413 "TEST 009 ALK Gate Array,ALP Gate Array Opcode Decode,RBUS-to-WBUS
(1) 1520 "TEST 00A RTMP0 Scratchpad Operational Test
(1) 1602 "TEST 00B IPR0 Scratchpad Operational Test
(1) 1684 "TEST 00C Q Register Data Integrity and Data Path Test
(1) 1813 "TEST 00D D Register Data Test
(1) 1925 "TEST 00E RTEMP Scratchpad Data Integrity
(1) 2028 "TEST 00F GPR Scratchpad Data Integrity
(1) 2132 "TEST 010 IPR Scratchpad Data Integrity
(1) 2236 "TEST 011 R Scratchpad Chip Select Test
(1) 2384 "TEST 012 R Scratchpad Write All Cells Test
(1) 2593 "TEST 013 RTEMP,GPR,IPR Scratchpad Address Integrity Test
(1) 2853 "TEST 014 DTYPE and ISIZE Decode VIA VBUS Test
(1) 3012 "TEST 015 RTEMP Scratchpad Byte Write,Word Write,Longword Write Test
(1) 3142 "TEST 016 M Scratchpad Basic Operational Test
(1) 3222 "TEST 017 M Scratchpad Data Integrity Test
(1) 3325 "TEST 018 M Scratchpad Address Integrity Test
(1) 3513 "TEST 019 MTEMP to RTEMP Cross Write Test
(1) 3619 "TEST 01A RTEMP to MTEMP Cross Write Test
(1) 3724 "TEST 01B ALP Gate Array Q Register Input Mux Test
(1) 3847 "TEST 01C ALU Zero and Sign Extend MBUS Data Test
(1) 4019 "TEST 01D ALU Functions Test
(1) 4662 "TEST 01E RNUM Register Data Integrity Test
(1) 4760 "TEST 01F Register Back Up Stack(RBS) and Pointer(RBSP) Tests
(1) 5529 "TEST 020 Default Write to M[TEMP0] Test
(1) 5723 "TEST 021 MTEMP Indexed by RNUM and RNUM+1 Test
(1) 5901 "TEST 022 SPA Gate Array Test #1
(1) 6091 "TEST 023 SPA Gate Array Test #2
(1) 6262 "TEST 024 SPA Gate Array Test #3
(1) 6446 "TEST 025 SPA Gate Array Test #4
(1) 6720 "TEST 026 SPA Gate Array Test #5
(1) 6936 "TEST 027 SPA Gate Array Test #6
(1) 7110 "TEST 028 ALU LOOP Flag Test
(1) 7204 "TEST 029 Test PSL Register
(1) 7339 "TEST 02A Test MISC, CCMISC, AND CCPSL Micro-Orders That Affect the PSL

```
(1)     7574        "TEST   02B     Test WCTRL Micro-Orders that affect the PSL Register
(1)     7742        "TEST   02C     ALK Gate Array Test #1
(1)     8059        "TEST   02D     ALK Gate Array Test #2
(1)     8409        "TEST   02E     ALK Gate Array Test #3
(1)     9023        "TEST   02F     ALK Gate Array Test #4
(1)     9346        "TEST   030     ALK Gate Array Test #5
(1)     9487        "TEST   031     ALK Gate Array Test #6
(1)     9936        "TEST   032     ALK Gate Array Test #7
(1)    10267        "TEST   033     Rotator Test  1 SBUS Data Integrity Test
(1)    10519        "TEST   034     Rotator Test  2 PLATCH Test
(1)    10625        "TEST   035     Rotator Test  3 SLATCH Test
(1)    10731        "TEST   036     Rotator Test  4 PLATCH 4:3 gets WBUS 1:0 Test
```

```
0000     1          .TITLE  ECKAB VAX-11/750 DPM MICRO DIAGNOSTICS
0000     2          .IDENT  /V09.01/
0000     3 ;
0000     4 ; COPYRIGHT (C) 1980,1983
0000     5 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     6 ;
0000     7 ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE  ONLY ON A SINGLE
0000     8 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000     9 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    10 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    11 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    12 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    13 ; REMAIN IN DEC.
0000    14 ;
0000    15 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    16 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    17 ; CORPORATION.
0000    18 ;
0000    19 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    20 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    21 ;
0000    22 ;
```

```
0000     24          .SBTTL  "DPM Revision History"
0000     25 ;++
0000     26 ; DPM MICRO-DIAGNOSTIC PACKAGE
0000     27 ;
0000     28 ; REVISION HISTORY
0000     29 ;
0000     30 ; 00-01 Tom Groetzinger 4-FEB-1980
0000     31 ;       Started file.
0000     32 ;
0000     33 ; 00-02 Dave Spain      27-FEB-1980
0000     34 ;       Added CS BUS INTEGRITY test.
0000     35 ;
0000     36 ; 00-03 Tom Groetzinger 12-MAR-1980
0000     37 ;       Added five tests.
0000     38 ;
0000     39 ; 00-04 Tom Groetzinger 17-MAR-1980
0000     40 ;       Added SRK,SRM callouts to some tests.
0000     41 ;
0000     42 ; 00-05 Tom Groetzinger 17-MAR-1980
0000     43 ;       Improved tests 7 and 16 to increase coverage due to Westboro feedback
0000     44 ;
0000     45 ; 00-06 Tom Groetzinger 17-MAR-1980
0000     46 ;       Debugged test 16 dueu to major rewrite
0000     47 ;
0000     48 ; 00-07 Tom Groetzinger 17-MAR-1980
0000     49 ;       Added 9 Super Rotator tests
0000     50 ;
0000     51 ; 01-08 Dave Spain       4-APR-80
0000     52 ;       Added 4 Micro-sequencer tests for offical release.
0000     53 ;
0000     54 ; 01-09 Bill Landry     4-APR-1980
0000     55 ;       Added the XB, PREFETCH and IRD tests from ECKAC.SRC so
0000     56 ;       manufacturing can run one tape for DPM.
0000     57 ;
0000     58 ; 00-10 Bill Landry     4-APR-1980
0000     59 ;       Let's save version 1.0 for our first SDC release.
0000     60 ;
0000     61 ; 00-11 Tom Groetzinger 7-APR-1980
0000     62 ;       Added 2 ALK Tests
0000     63 ;
0000     64 ; 00-12 Tom Groetzinger 8-APR-1980
0000     65 ;       Deleted last 2 D Register outputs from ALK Test #6
0000     66 ;
0000     67 ; 00-13 Tom Groetzinger 14-APR-1980
0000     68 ;       Added D Register output to ALK test #6
0000     69 ;
0000     70 ; 00-14 Dave Spain      25-APR-1980
0000     71 ;       Changed mask on the micro-sequencer tests to ignore bit 13 of the
0000     72 ;       Control Store address.
0000     73 ;
0000     74 ; 00-15 Dave Spain      28-APR-1980
0000     75 ;       Fixed bug in ALK Tests 6 and 7 by replacing VSIZE/1 micro order with a
0000     76 ;       SIZE[LONG] macro in micro words 6 and 3 respectively.
0000     77 ;
0000     78 ; 00-16 Dave Spain      29-APR-1980
```

```
0000     79 ;       Changed mask on CS address for Instruction Address Decode test from
0000     80 ;       3FFF to 1FFF to ignore bit 13 of Control Store address.
0000     81 ;
0000     82 ; 00-17 Dave Spain      30-APR-1980
0000     83 ;       Added CCS module to error callout for Control Store Bus Integrity test.
0000     84 ;
0000     85 ; 00-18 Dave Spain       9-MAY-1980
0000     86 ;       Added SRM gate array to error callout routine in RTMP0 and RTEMP
0000     87 ;       Sratchpad tests.
0000     88 ;
0000     89 ; 00-19 Bill Landry     10-JUN-1980
0000     90 ;       Added RDM tests to beginning of tape
0000     91 ;
0000     92 ; 01-00 Bill Landry/Dave Spain  13-JUN-1980
0000     93 ;       Added macro file to title section and compatibility mode IRD tests
0000     94 ;       then submitted to RELEASE ENGINERRING.
0000     95 ;
0000     96 ; 01-01 Dave Spain      19-JUN-1980
0000     97 ;       Fixed prefetch incrementer test to clear out cache data after every
0000     98 ;       prefetch. This will prevent the test from mistaking bad addresses from
0000     99 ;       good ones.
0000    100 ;
0000    101 ; 01-02 Dave Spain       2-JUL-1980
0000    102 ;       Added CM Second IRDX tests to file.
0000    103 ;
0000    104 ; 01-03 Dave Spain      18-AUG-1980
0000    105 ;       Added CM branch test to file.
0000    106 ;
0000    107 ; 01-04 Dave Spain      10-SEPT-1980
0000    108 ;       Fixed "DEFAULT MTEMP0 TEST"
0000    109 ;
0000    110 ; 01-05 Bill Landry     11-SEPT-1980
0000    111 ;       Revised RDM to CMI tests
0000    112 ;
0000    113 ; 02-00 Dave Spain      28-SEPT-1980
0000    114 ;       Added rotator tests to file and submitted to SDC.
0000    115 ;
0000    116 ; 02-01 Dave Spain      28-OCT-1980
0000    117 ;       Added QV flag to tests to be run by manufacturing.
0000    118 ;
0000    119 ; 02-02 Bill Landry     10-NOV-1980
0000    120 ;       Added tests for TOK and SAC gate arrays.
0000    121 ;
0000    122 ; 02-03 Bill Landry     13-NOV-1980
0000    123 ;       Added test of REI CHECK
0000    124 ;
0000    125 ; 03-00 Dave Spain      17-NOV-1980
0000    126 ;       Added rotator tests 23-28 and released to SDC.
0000    127 ;
0000    128 ; 03-01 Dave Spain      10-DEC-1980
0000    129 ;       Added rotator tests 29-42.
0000    130 ;
0000    131 ; 03-02 Bill Landry     11-DEC-1980
0000    132 ;       Fixed bug in CS ADDRESS test
0000    133 ;
```

```
0000    134 ; 03-03 Dave Spain      12-DEC-1980
0000    135 ;       Enhanced Rotator Test 1 to call out ALP slices.
0000    136 ;
0000    137 ; 03-04 Bill Landry     16-DEC-1980
0000    138 ;       Added DSIZE ROM tests
0000    139 ;
0000    140 ; 03-05 Bill Landry     31-DEC-1980
0000    141 ;       Last great act of 80. Separated DPM and RDM tests into two files.
0000    142 ;
0000    143 ; 04-00 Bill Landry     5-JAN-1981
0000    144 ;       Submitted to RELEASE ENGINERRING
0000    145 ;
0000    146 ; 04-01 Bill Landry     14-JAN-1981
0000    147 ;       Revised INTERVAL TIMER INTERRUPT TEST to fix bug caused by INT PEND
0000    148 ;       ECO
0000    149 ;
0000    150 ; 04-02 Dave Spain      16-JAN-1981
0000    151 ;       Fixed error callout in CS Parity Error, XB, and Prefetch Incrementer
0000    152 ;       tests and added seven new rotator tests.
0000    153 ;
0000    154 ; 04-03 Bill Landry     4-FEB-1981
0000    155 ;       Added tests for CCC gate array
0000    156 ;
0000    157 ; 04-04 Dave Spain      9-FEB-1981
0000    158 ;       Changed the names of rotator tests 23-33.
0000    159 ;
0000    160 ; 04-05 Bill Landry     24-FEB-1981
0000    161 ;       Added tests for DPM discrete branching logic.
0000    162 ;
0000    163 ; 04-06 Bill Landry     27-FEB-1981
0000    164 ;       Added tests for branching logic in PHB gate array.
0000    165 ;
0000    166 ; 05-00 Dave Spain       1-MAR-1981
0000    167 ;       Added 5 SRK status rotator tests and submitted to SDC.
0000    168 ;
0000    169 ; 05-01 Dave Spain      25-MAR-1981
0000    170 ;       Fixed a loop on test bug in the Alu Functions Test by adding a LDFIELD
0000    171 ;       pseudo-op that reinitializes the ROT field in Subtest 1.
0000    172 ;
0000    173 ; 05-02 Bill Landry     25-MAR-1981
0000    174 ;       Fixed bug in BUT/NO.FPA test.  I was branching on the wrong conditions.
0000    175 ;
0000    176 ; 05-03 Dave Spain      14-APR-1981
0000    177 ;       Cleaned up documentation for R Scratchpads Write All Cells Test.
0000    178 ;
0000    179 ; 06-00 Bill Landry     2-MAY-1981
0000    180 ;       Added SPA branching, CON registers, TU58 registers, and CCS PARITY
0000    181 ;       tests.  Submitted to Release Engineering.
0000    182 ;
0000    183 ; 06-01 Dave Spain      17-JUN-1981
0000    184 ;       Replaced the old IRD1 Instruction Address Decode test with two new
0000    185 ;       tests that toggle the PSL's FPD bit and can handle an FPA.
0000    186 ;
0000    187 ; 06-02 Bill Landry     19-AUG-1981
0000    188 ;       Fixed bugs created by introduction of FPA option.
```

```
0000   189 ;       Replaced all references to COMET with the word CPU.
0000   190 ;       Fixed typing error in vbus data of CPU CLOCK TEST.
0000   191 ;       Submitted to RELEASE ENGINEERING.
0000   192 ;
0000   193 ; 06-03 Bill Landry     24-SEPT-1981
0000   194 ;       Fixed potential clock problem with FPA in Z BIT TEST 1.
0000   195 ;
0000   196 ; 06-04 Bill Landry     9-OCT-1981
0000   197 ;       Added changes for ECO L0002-TW002
0000   198 ;
0000   199 ; 07-00 Bill Landry     31-OCT-1981
0000   200 ;       Removed changes for ECO L0002-TW002.  ECO is on hold.
0000   201 ;       Added tests for FPA to CPU interface logic
0000   202 ;       Moved WCS test to MIC tape
0000   203 ;       Submitted to RELEASE ENGINEERING.
0000   204 ;
0000   205 ; 07-01 Bill Landry     13-NOV-1981
0000   206 ;       Stopped release of 7.0
0000   207 ;       Added FPA TRAP LOGIC TEST
0000   208 ;       Enhanced error messages for FPA
0000   209 ;       Resubmitted to RELEASE ENGINEERING
0000   210 ;
0000   211 ; 07-02 Bill Landry     20-NOV-1981
0000   212 ;       Stopped release of 7.1
0000   213 ;       Enhanced FPA ENABLE/DISABLE TEST to check for MSEQ INIT
0000   214 ;       Resubmitted to RELEASE ENGINEERING
0000   215 ;
0000   216 ; 07-03 Dave Shull      26-Mar-1982
0000   217 ;       Added the ERRLOOP pseudo to the Rotator Extract tests which were
0000   218 ;       missing these and couldn't be looped on.
0000   219 ;
0000   220 ; 07-04 Dave Shull      05-Apr-1982
0000   221 ;       Removed all BEGINSA and ENDSA pseudo's.
0000   222 ;       Submitted to RELEASE ENGINEERING
0000   223 ;
0000   224 ; 07-05 Dave Shull      21-May-1982
0000   225 ;       Add a subtest to the Micro Sequencer Stack Test that will test bits
0000   226 ;       11 and 12 of the micro stack.
0000   227 ;
0000   228 ; 07-06 Dave Shull      24-May-1982
0000   229 ;       Changed documentation in the CM Odd Address BUT Test to reflect the
0000   230 ;       ECO L0002-TW002
0000   231 ;
0000   232 ; 08-00 Dave Shull      18-Jun-1982
0000   233 ;       Added 10 new tests.  Two of those tests are an addition to the existing
0000   234 ;       IRD1 rom tests.  These 2 new tests check the signal ROM OS INH H.
0000   235 ;       The other 8 new tests check the validity of the IRDX ROMs.  They check
0000   236 ;       both the address and the signal ROM OS INH H.
0000   237 ;
0000   238 ; 08-01 Dave Shull      22-Jun-1982
0000   239 ;       Added a subtest to the Register Backup Stack test that will check the
0000   240 ;       bit that records Increment/Decrement of the GPR.
0000   241 ;
0000   242 ; 09-00 Dave Shull      03-Nov-1982
0000   243 ;       Added two new Tests for coverage the new PCS (L0008) module.
```

```
0000   244 ;
0000   245 ; 09-01 Dave Smith      15-May-1983
0000   246 ;       Correct DCS in ALK test 3,6,7. Disabled ALP's on the WBUS.
0000   247 ;--
0000     1          .SBTTL  ""
```

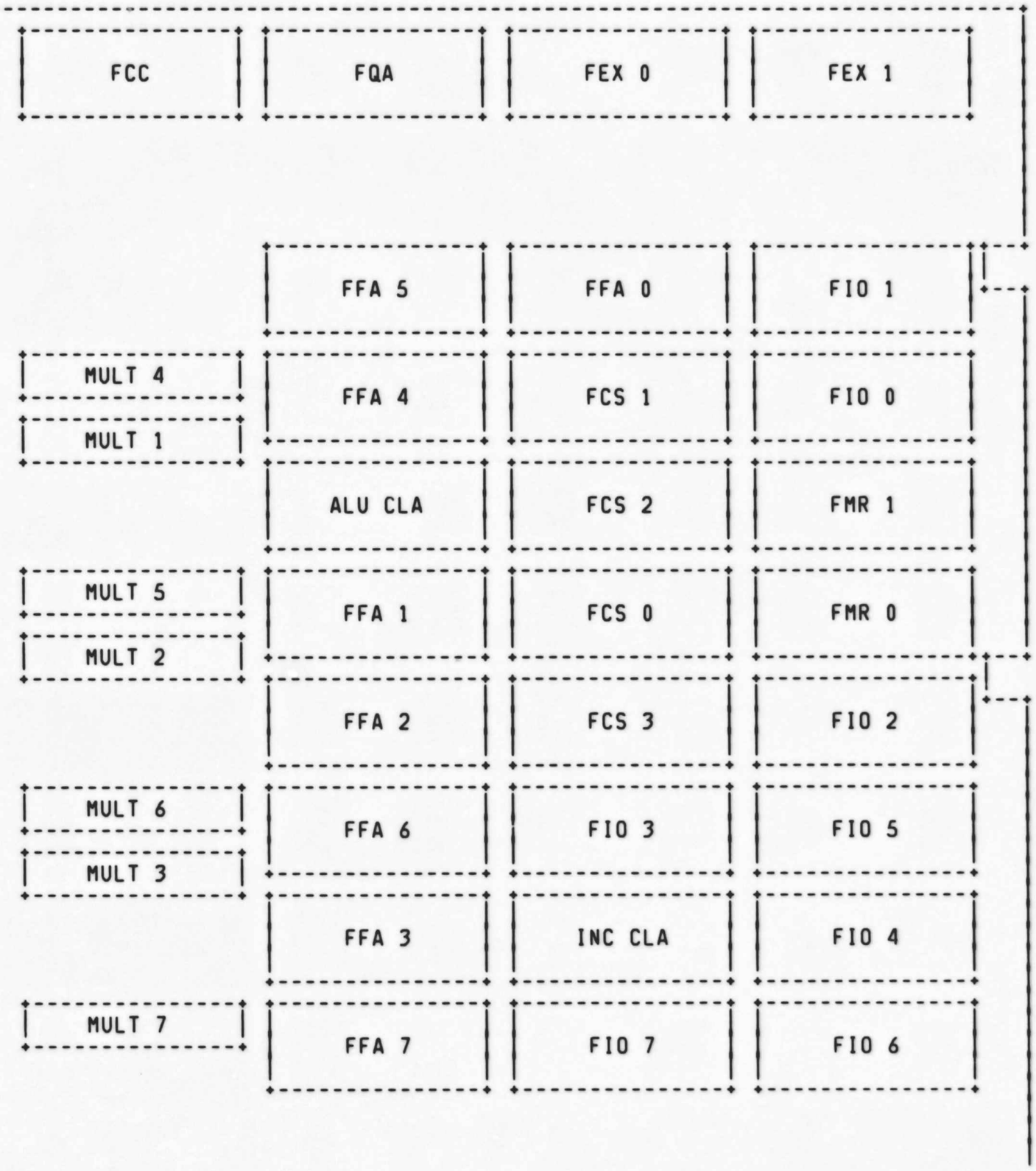
```
0000     3          .SBTTL  "DIAGNOSTIC MICROCODE MACROS"
0000     4 ;++
0000     5 ; Revision History
0000     6 ;
0000     7 ; 28    Dave Spain      22-Jun-82
0000     8 ;       Removed LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
0000     9 ;       Decided not to use them.
0000    10 ;
0000    11 ; 27    Dave Spain      15-Jun-82
0000    12 ;       Added LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
0000    13 ;
0000    14 ; 26    Dave Spain      19-May-82
0000    15 ;       Added M[]_FPA macro.
0000    16 ;
0000    17 ; 25    Rich Lewis      23-Feb-82
0000    18 ;       Added PSL_WB, M[]_R[].OR.NIB0(M), WB_FPA CCR, WB_FPA TCR, Q_R[].ASL.P,
0000    19 ;       RDM_M[].XZ.OR.R[], FPA_WB macros
0000    20 ;
0000    21 ; 24    Dave Spain      02-Feb-82
0000    22 ;       Added FPA_R[]+M[] macro.
0000    23 ;
0000    24 ; 23    Bill Landry     22-OCT-81
0000    25 ;       Added PSL_RDM macro
0000    26 ;
0000    27 ; 22    Dave Spain       5 Oct 81
0000    28 ;       Added RDM_FPA and FPA_M[] FPA_RDM macros.
0000    29 ;
0000    30 ; 21    Bill Landry     29 Sept 81
0000    31 ;       Added FPA macro, FPA_M[] FPA_R[].
0000    32 ;
0000    33 ; 20    Dave Spain      14 Aug 81
0000    34 ;       Added the TESTFPA [] macro.
0000    35 ;
0000    36 ; 19    Dave Spain      18 July 81
0000    37 ;       Added the following FPA macros, FPA_RDM and FPA_R[].
0000    38 ;
0000    39 ; 18    Dave Spain      24 Sept 80
0000    40 ;       Added RDM_Q macro. This macro uses the ALU.
0000    41 ;
0000    42 ; 17    Bill Landry     23 Sept 80
0000    43 ;       Added macro VA_VA-ZLIT0[] for testing memory in decending order.
0000    44 ;
0000    45 ; 16    Dave Spain      27 August 80
0000    46 ;       Changed RDM_Q macro to RDM_Q Q_D. This more accurately reflects macro.
0000    47 ;
0000    48 ; 15    Bill Landry     4 June 80
0000    49 ;       Deleted BUS/PRB.RD.PTE from macro CLRTB.VA_VA and CLRCH.VA_VA to
0000    50 ;       satisfy validity checks.
0000    51 ;
0000    52 ; 14    Dave Spain      23 May 80
0000    53 ;       Added BUS/PRB.RD.PTE to the clear cache and clear TB macros for Mr.
0000    54 ;       Bill.
0000    55 ;
0000    56 ; 13    Dave Spain      14 Apr 80
0000    57 ;       Added SIZE[] or VSIZE/1 to some macros. This allows version 65 of
```

```
0000     58 ;       the COMET micro-code defines to run.
0000     59 ;
0000     60 ; 12    Bill Landry       1 Apr 80
0000     61 ;       Added BUS/PRB.RD micro order to RDM_TB macro to compensate for timing
0000     62 ;       problem in ADD gate array.  This is not an APRIL FOOL!
0000     63 ;
0000     64 ; 11    Dave Spain       27 Mar 80
0000     65 ;       Fixed TB macro's to include MSRC/VA to prevent validity failure.
0000     66 ;
0000     67 ; 10    Bill Landry      19 Mar 80
0000     68 ;       Added Q_R[].RL.P and VA_Q.OR.R[]
0000     69 ;
0000     70 ; 09    Bill Landry      19 Mar 80
0000     71 ;       Deleted TB_R[].RL.P.OR.D
0000     72 ;
0000     73 ; 08    Bill Landry      18 Mar 80
0000     74 ;       Added VA_R[].RL.P
0000     75 ;
0000     76 ; 07    Bill Landry      07 Mar 80
0000     77 ;       Added Q_M[].RL.PTE
0000     78 ;
0000     79 ; 06    Dave Spain       06 Mar 80
0000     80 ;       Changed TB_R[].RL.P macro to TB_R[].RL.P.OR.D.
0000     81 ;
0000     82 ; 05    Dave Spain       03 Mar 80
0000     83 ;       Added TB_R[].RL.P macro for Bill Landry.
0000     84 ;
0000     85 ; 04    Dave Spain       22 Feb 80
0000     86 ;       Added CLRTB.VA_R[] macro for Bill Landry.
0000     87 ;
0000     88 ; 03    Dave Spain       30 Jan 80
0000     89 ;       Added Tom Groetzinger's macro definitions.
0000     90 ;
0000     91 ; 02    Dave Spain       30 Jan 80
0000     92 ;       Added Tony Vezza's macro definitions.
0000     93 ;
0000     94 ; 01    Dave Spain       29 Jan 80
0000     95 ;       Macros initiated.
0000     96 ;--
0000     97
0000     98          .SBTTL  "       RDM Control File Macros"
0000     99
0000    100 ;STROBE.V.BUS            "RDMCF0/V.STROBE"
0000    101 ;DCS.ADDR_0              "RDMCF1/DCS.ADDR.CLR"
0000    102 ;STOP.CLOCK              "RDMCF2/STOP.CPU.CLOCK"
0000    103 ;LATCH.CS.ADDR           "RDMCF3/DIS.STROBE.CS.A"
0000    104 ;RDM_WB                  "RDMCF4/H.REG.FROM.WB"
0000    105 ;WB_RDM                  "RDMCF5/WB.FROM.D.REG"
0000    106 ;RDM_CMI                                "RDMCF6/STROBE.CMI"
0000    107 ;INH.CLOCK.SA            "RDMCF7/INH.SA.CLOCK"
0000    108
0000    109          .SBTTL  "       Diagnostic Macros"
0000    110
0000    111 ;CLOCK.EXT                              "CLKX/XTND"
0000    112 ;CLRCH.VA_RDM                           "WB_RDM,WCTRL/CLRCH.VA_WB,BUS/PRB.RD.PTE"
```

```
0000   113 ;CLRCH.VA_VA                    "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRCH.VA_WB
0000   114 ;CLRTB.VA_D                     "RSRC/ZERO,MUX/D.R1,ALU/OR,WCTRL/CLRTB.VA_WB,
0000   115 ;                                       BUS/PRB.RD.PTE"
0000   116 ;CLRTB.VA_RDM                   "WB_RDM,WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
0000   117 ;CLRTB.VA_R[]                   "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/CLRTB.VA_WB,
0000   118 ;                                       BUS/PRB.RD.PTE"
0000   119 ;CLRTB.VA_VA                    "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRTB.VA_WB
0000   120 ;CLRTB.VA_WB                    "WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
0000   121 ;D_LONLIT                       "RSRC/LONLIT,ALPCTL/WX_D_R.Q_D"
0000   122 ;D_M[]_LONLIT                   "D_LONLIT,MSRC/a1"
0000   123 ;D_VA_0                         "RSRC/ZERO,ROT/ZERO,MUX/R.S,ALU/OR,DQ1/D_WX,
0000   124 ;                                       WCTRL/VA_WB"
0000   125 ;FPA_M[] FPA_RDM                       "MSRC/a1,WB_RDM,FPA/FPA_MBUS.FPA_WBUS"
0000   126 ;FPA_M[] FPA_R[]                       "FPA/FPA_MBUS.FPA_WBUS,MSRC/a1,RSRC/a2,MUX/R
0000   127 ;                                       ROT/ZERO,ALU/OR"
0000   128 ;FPA_RDM                               "WB_RDM,FPA/FPA_DATA.WBUS"
0000   129 ;FPA_R[]                               "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,FPA/FPA_DAT
0000   130 ;FPA_R[]+M[]                   "RSRC/a1,MSRC/a2,MUX/M.R1,ALU/A+B+CI,ALUCI/ZERO,FPA/
0000   131 ;FPA_WB                        "FPA/FPA_DATA.WBUS"
0000   132 ;MEMSCAR_R[]                   "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/MEMSCAR_WB"
0000   133 ;M[]_FPA                               "MSRC/a1,SPW/MLONG,FPA/WBUS_FPA"
0000   134 ;M[]_LONLIT                   "MSRC/a1,SPW/MLONG,ALPCTL/WX_R.Q_D,RSRC/LONLIT"
0000   135 ;M[]_R[].OR.NIB0(M)           "MSRC/a1,RSRC/a2,SPW/MLONG,ROT/GETNIB,ALU/OR,
0000   136 ;                                       MUX/R.S,DQ1/NOP"
0000   137 ;M[]_RDM                               "WB_RDM,SPW/MLONG,MSRC/a1"
0000   138 ;PC_PC+RDM                     "WCTRL/PC_PC+WB,WB_RDM"
0000   139 ;PC_PC+4 MB_XB                 "MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,SIZE[LONG]"
0000   140 ;PC_PC+4 RDM_XB                "RSRC/ZERO,MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,
0000   141 ;                                       MUX/M.R1,ALU/OR,RDM_WB,SIZE[LONG]"
0000   142 ;PC_RDM                        "WB_RDM,WCTRL/PC_WB"
0000   143 ;PSL_RDM                               "CCPSL/PSL_WB.CCBR_ALUS,WB_RDM"
0000   144 ;PSL_WB                        "CCPSL/PSL_WB.CCBR_ALUS"
0000   145 ;Q_D WX_R                      "ALPCTL/WX_R.Q_D"
0000   146 ;Q_LONLIT                      "RSRC/LONLIT,ALPCTL/WX_S.Q_R"
0000   147 ;Q_M[].RL.PTE                  "ALPCTL/WX_Q_S,MSRC/a1,ROT/RL.MM.PTE"
0000   148 ;Q_R[].AND.Q                   "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/AND"
0000   149 ;Q_R[].ASL.P                   "ALPCTL/WX_Q_S,RSRC/a1,ROT/ASL.R.P"
0000   150 ;Q_R[].OR.Q                    "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/OR"
0000   151 ;Q_R[].RL.P                    "ALPCTL/WX_Q_S,RSRC/a1,ROT/RL.RR.P"
0000   152 ;RDM_ALUF                      "ALPCTL/WB_ALUF,RDM_WB"
0000   153 ;RDM_FPA                               "FPA/WBUS_FPA,RDM_WB"
0000   154 ;RDM_LONLIT                   "WB_LONLIT,RDM_WB"
0000   155 ;RDM_LONLIT.Q_M               "RSRC/LONLIT,ALPCTL/WX_R.Q_M,RDM_WB"
0000   156 ;RDM_M[]                               "MSRC/a1,ROT/ZERO,MUX/M.S,ALU/OR,RDM_WB"
0000   157 ;RDM_M[].OR.R[]               "WB_M[a1].OR.R[a2],RDM_WB"
0000   158 ;RDM_M[]-R[]                  "WB_M[a1]-R[a2],RDM_WB"
0000   159 ;RDM_M[].XZ.OR.R[]           "ALU/A+B+CI,MUX/R.S,ROT/XZ.MM,MSRC/a1,RSRC/a2,
                                        RDM_WB"
0000   160 ;
0000   161 ;RDM_PC                        "MSRC/PC,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
0000   162 ;RDM_PCBACK                    "RSRC/ZERO,MSRC/PCBACK,MUX/M.R1,ALU/OR,RDM_WB"
0000   163 ;RDM_Q                         "RSRC/ZERO,MUX/R.Q,ALU/OR,RDM_WB"
0000   164 ;RDM_Q Q_D                     "ALPCTL/WX_Q.Q_D,RDM_WB"
0000   165 ;RDM_R[]                               "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,RDM_WB"
0000   166 ;RDM_TB                        "WB_M[TB],RDM_WB,BUS/PRB.RD,SIZE[LONG]"
0000   167 ;RDM_VA                        "MSRC/VA,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
```

```
0000   168 ;RNUM_LONLIT                    "ALPCTL/WX_D_R.Q_M,MSRC/RNUM_WBUS,RSRC/LONLIT"
0000   169 ;R[]_RDM                                "WB_RDM,SPW/RLONG,RSRC/a1"
0000   170 ;R[].DT_Q                       "RSRC/a1,D_Q Q_D,SPW/RSIZE,VSIZE/1"
0000   171 ;R[]_VA                        "RSRC/a1,MSRC/VA,ROT/ZERO,MUX/M.S,ALU/OR,SPW/RLONG"
0000   172 ;R[]_WB                        "RSRC/a1,SPW/RLONG"
0000   173 ;TB_RDM                        "WCTRL/TB_WB,WB_RDM,MSRC/VA"
0000   174 ;TB_WB                         "WCTRL/TB_WB,MSRC/VA"
0000   175 ;TESTFPA []                    "TESTFPA/a1"
0000   176 ;VA_PC+LONLIT+I.PC_PC+I        "RSRC/LONLIT,MSRC/XB.PC_PC+I,ROT/ZERO,MUX/R.S,ALU/OR
0000   177 ;                                      ISTRM/ISIZE_DSIZE,WCTRL/VA_PC+I+W.PC_PC+I,
0000   178 ;                                      SIZE[LONG]"
0000   179 ;VA_Q.OR.R[]                    "WCTRL/VA_WB,RSRC/a1,ALU/OR,MUX/R.Q"
0000   180 ;VA_RDM                        "WB_RDM,WCTRL/VA_WB"
0000   181 ;VA_R[].RL.P                   "ALPCTL/WX_S,ROT/RL.RR.P,RSRC/a1,WCTRL/VA_WB"
0000   182 ;VA_VA-ZLIT0[]                 "WCTRL/VA_WB,LIT/LITRL,LITRL/a1,ROT/ZLIT0,MSRC/VA,
0000   183 ;                                      MUX/M.S,ALU/A-B-CI"
0000   184 ;VA_WB                         "WCTRL/VA_WB"
0000   185 ;WB_FPA CCR                    "FPA/WBUS_FPA.CC"
0000   186 ;WB_FPA TCR                    "WCTRL/FPTCR"
0000   187 ;WB_LONLIT                     "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
0000   188 ;WB_LONLIT.Q_M                 "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
0000   189 ;WB_M[]+R[]                    "ALU/A+B+CI,ALUCI/ZERO,MUX/M.R1,MSRC/a1,RSRC/a2"
0000   190 ;WB_R[].Q_D                    "RSRC/a1,ALPCTL/WX_R.Q_D"
0000   191 ;WDR_R[]                              "WCTRL/WDR_WB,ALU/OR,MUX/R.S,RSRC/a1,ROT/ZER
0000   192 ;WRITE.LONG R[]                "BUS/WRITE.LNG,WCTRL/WDR_WB.UR,RSRC/a1,ROT/ZERO,
0000   193 ;                                      MUX/R.S,ALU/OR,SIZE[LONG]"
0000   194         .SBTTL  ""
0000   195         .SBTTL  "MODULE GATE ARRAY MAPS"
```

```
                                .SBTTL  "       L0001 MULTIPLIER AND GATE ARRAY LOCATION"
```

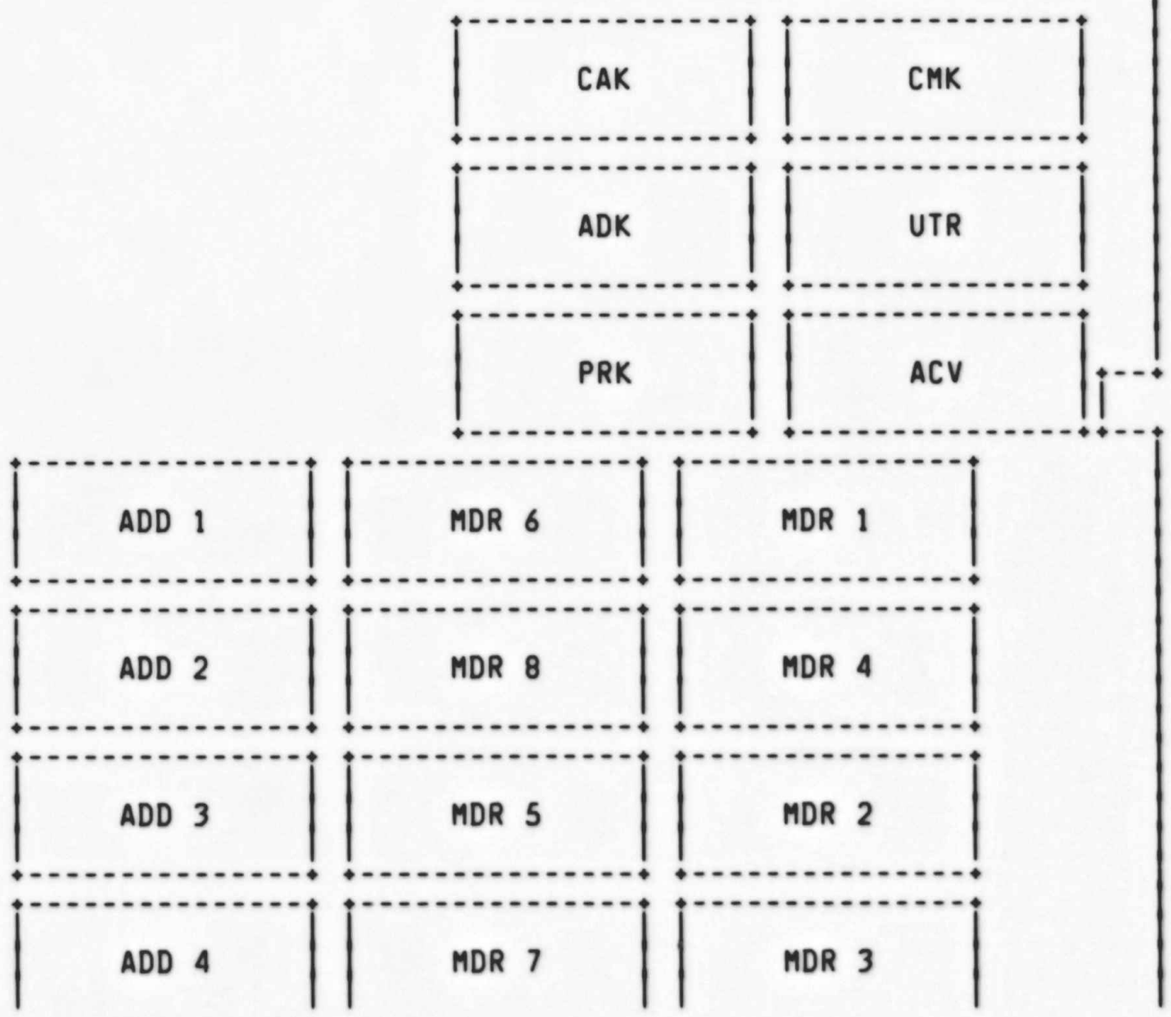


```
                    0000   252 ;+-----------------------------------------------------------------------------+
```

```
0000          .SBTTL  "       L0002 GATE ARRAY LOCATION"
0000   ;+-----------------------------------------------------------------------------+
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                  +---------------+ +---------------+                        |
0000   ;|                  |               | |               |                        |
0000   ;|                  |      PHB      | |      MSQ      |                        |
0000   ;|                  |               | |               |                        |
0000   ;|                  +---------------+ +---------------+                        |
0000   ;|                  +---------------+ +---------------+                    +---+
0000   ;|                  |               | |               |                    |   |
0000   ;|                  |      CCC      | |      SAC      |                    +---+
0000   ;|                  |               | |               |                        |
0000   ;|                  +---------------+ +---------------+                        |
0000   ;|  +---------------+ +---------------+ +---------------+                        |
0000   ;|  |               | |               | |               |                        |
0000   ;|  |      SPA      | |      SRK      | |      IRD      |                        |
0000   ;|  |               | |               | |               |                        |
0000   ;|  +---------------+ +---------------+ +---------------+                        |
0000   ;|  +---------------+ +---------------+ +---------------+                        |
0000   ;|  |               | |               | |               |                        |
0000   ;|  |      TOK      | |      CLA      | |      ALK      |                        |
0000   ;|  |               | |               | |               |                        |
0000   ;|  +---------------+ +---------------+ +---------------+                        |
0000   ;|                  +---------------+ +---------------+                        |
0000   ;|                  |               | |               |                        |
0000   ;|                  |      ALP 1    | |      ALP 5    |                        |
0000   ;|                  |               | |               |                        |
0000   ;|                  +---------------+ +---------------+                    +---+
0000   ;|                  +---------------+ +---------------+                    |   |
0000   ;|                  |               | |               |                    +---+
0000   ;|                  |      ALP 2    | |      ALP 6    |                        |
0000   ;|                  |               | |               |                        |
0000   ;|                  +---------------+ +---------------+                        |
0000   ;|                  +---------------+ +---------------+                        |
0000   ;|                  |               | |               |                        |
0000   ;|                  |      ALP 3    | |      ALP 7    |                        |
0000   ;|                  |               | |               |                        |
0000   ;|                  +---------------+ +---------------+                        |
0000   ;|                  +---------------+ +---------------+                        |
0000   ;|                  |               | |               |                        |
0000   ;|                  |      ALP 4    | |      ALP 8    |                        |
0000   ;|                  |               | |               |                        |
0000   ;|                  +---------------+ +---------------+                        |
0000   ;|  +---------------+ +---------------+ +---------------+ +---------------+        |
0000   ;|  |               | |               | |               | |               |        |
0000   ;|  |      SRM 1    | |      SRM 2    | |      SRM 3    | |      SRM 4    |        |
0000   ;|  |               | |               | |               | |               |        |
0000   ;|  +---------------+ +---------------+ +---------------+ +---------------+        |
```

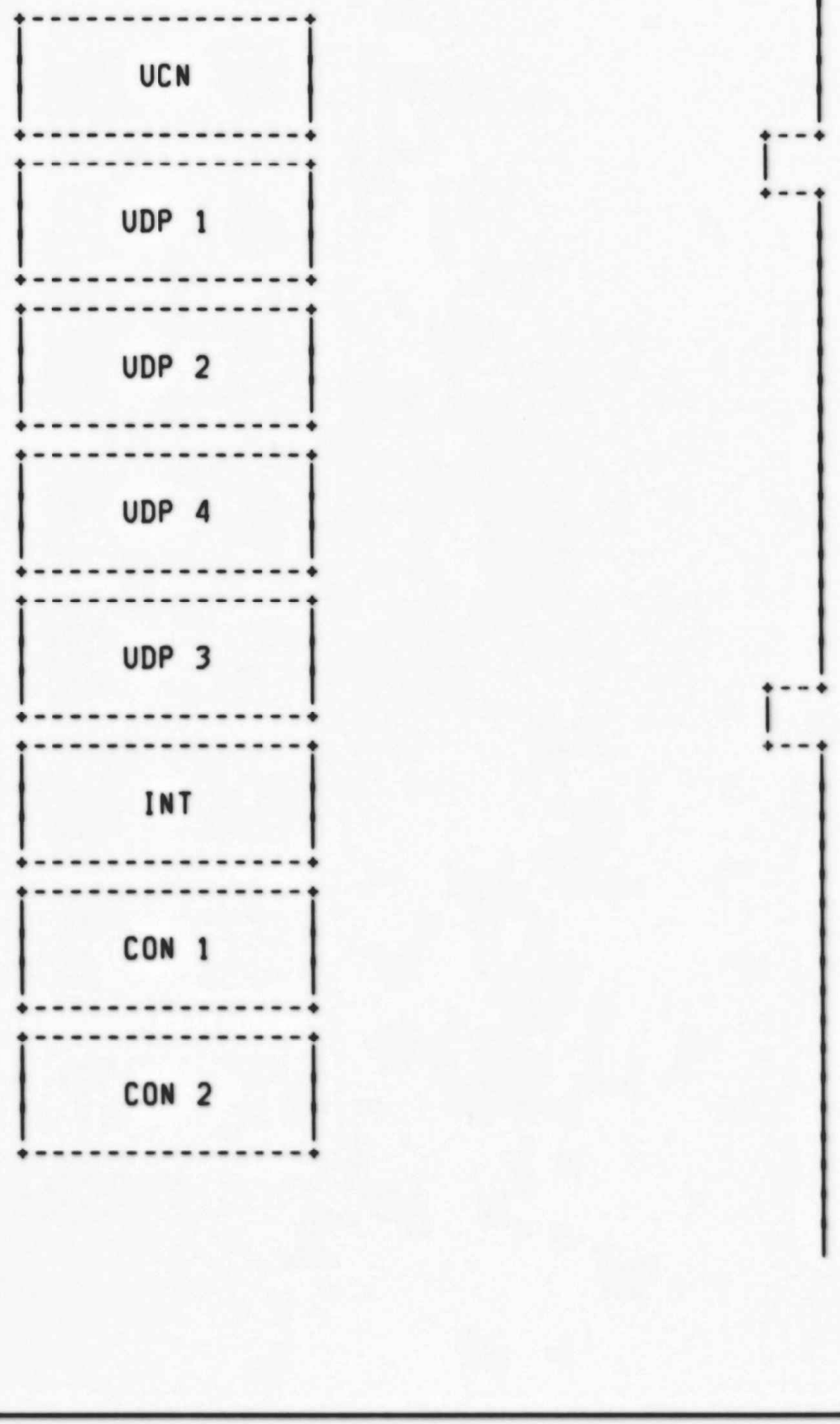
```
                0000   309 ;+----------------------------------------------------------------------------+
```

```
                        0000    311          .SBTTL  "       L0003 GATE ARRAY LOCATION"
                        0000    312 ;+-----------------------------------------------------------------------------+
                        0000    313 ;|                                                                             |
                        0000    314 ;|                                                                             |
                        0000    315 ;|                                                                             |
                        0000    316 ;|                                                                             |
                        0000    317 ;|                                                                             |
                        0000    318 ;|                                                                             |
                        0000    319 ;|                                                                             |
                        0000    320 ;|                                                                             |
                        0000    321 ;|                                                                             |
                        0000    322 ;|                                                                             |
                        0000    323 ;|                                                                             |
                        0000    324 ;|                                                                             |
                        0000    325 ;|                                                                             |
                        0000    326 ;|                                                                             |
                        0000    327 ;|                                                                             |
                        0000    328 ;|                                                                             |
                        0000    329 ;|                                                                             |
                        0000    330 ;|                                                                             |
                        0000    331 ;|                                                                             |
                        0000    332 ;|                                                                             |
                        0000    333 ;|                                                                             |
                        0000    334 ;|                                                                             |
                        0000    335 ;|                                                                             |
                        0000    336 ;|                                                                             |
                        0000    337 ;|                                                                             |
                        0000    338 ;|                                                                             |
                        0000    339 ;|                                                                             |
                        0000    340 ;|                                                                             |
                        0000    341 ;|                                                                             |
                        0000    342 ;|                                                                             |
                        0000    343 ;|                                                                             |
                        0000    344 ;|                                                                             |
                        0000    345 ;|                                                                             |
                        0000    346 ;|                                                                             |
                        0000    347 ;|                                                                             |
                        0000    348 ;|                                                                             |
                        0000    349 ;|                                                                             |
                        0000    350 ;|                                                                             |
                        0000    351 ;|                                                                             |
                        0000    352 ;|                                                                             |
                        0000    353 ;|                                                                             |
                        0000    354 ;|                                                                             |
                        0000    355 ;|                                                                             |
                        0000    356 ;|                                                                             |
                        0000    357 ;|                                                                             |
                        0000    358 ;|                                                                             |
                        0000    359 ;|                                                                             |
                        0000    360 ;|                                                                             |
                        0000    361 ;|                                                                             |
                        0000    362 ;|                                                                             |
                        0000    363 ;|                                                                             |
                        0000    364 ;|                                                                             |
                        0000    365 ;|                                                                             |
```



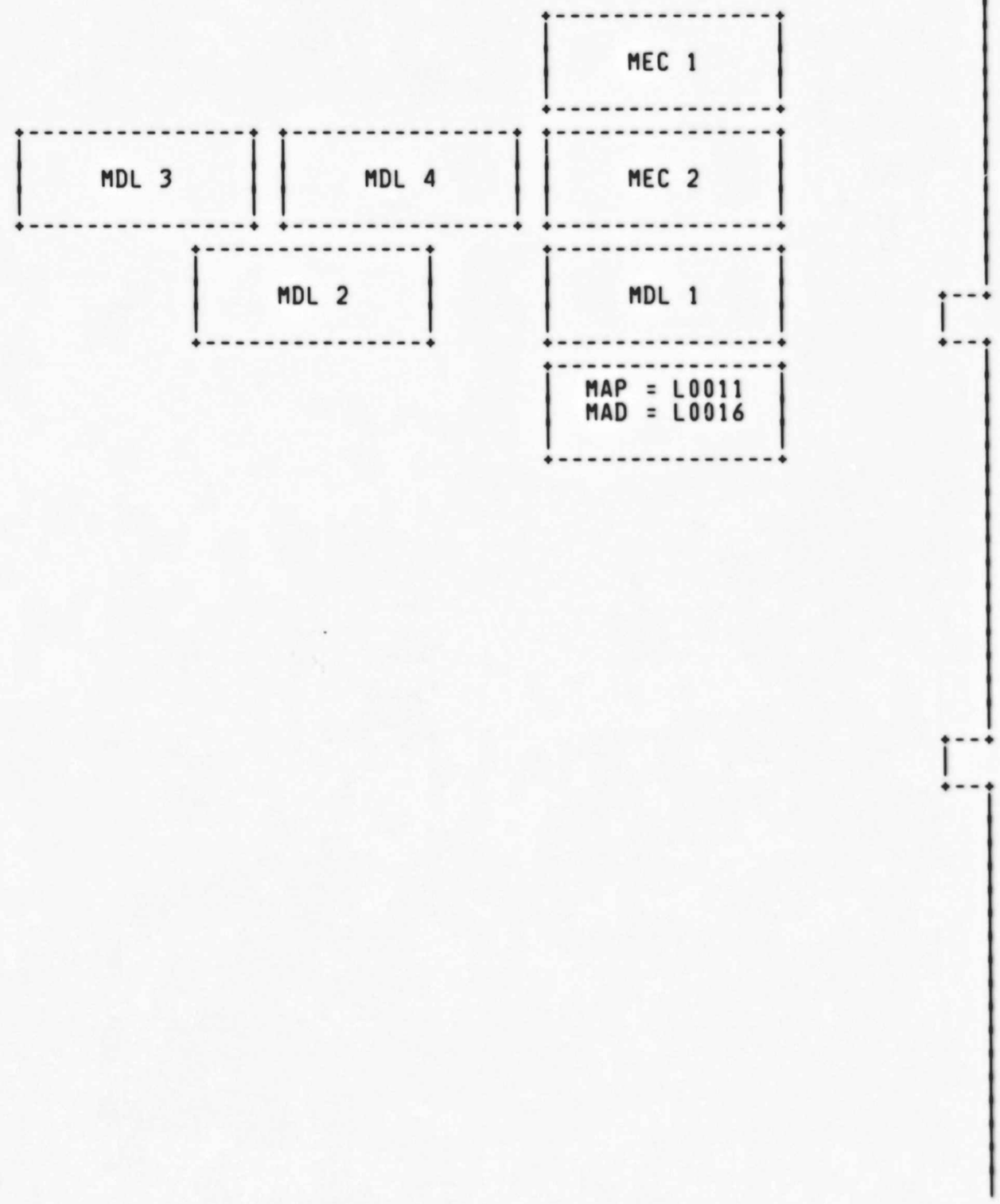
```
                        0000   366 ;|               +---------------+ +---------------+ +---------------+          |
                        0000   367 ;+-------------------------------------------------------------------------------+
```

```
                        0000   369          .SBTTL  "       L0004 GATE ARRAY LOCATION"
                        0000   370 ;+-----------------------------------------------------------------------------+
                        0000   371 ;|                                                                             |
                        0000   372 ;|                                                                             |
                        0000   373 ;|                                                                             |
                        0000   374 ;|                                                                             |
                        0000   375 ;|                                                                             |
                        0000   376 ;|                                                                             |
                        0000   377 ;|                                                                             |
                        0000   378 ;|                                                                             |
                        0000   379 ;|                                                                             |
                        0000   380 ;|                                                                             |
                        0000   381 ;|                                                                             |
                        0000   382 ;|                                                                             |
                        0000   383 ;|                                                                             |
                        0000   384 ;|                                                                             |
                        0000   385 ;|                                                                             |
                        0000   386 ;|                                                                             |
                        0000   387 ;|                                                                             |
                        0000   388 ;|                                                                             |
                        0000   389 ;|                                                                             |
                        0000   390 ;|                                                                             |
                        0000   391 ;|                                                                             |
                        0000   392 ;|                                                                             |
                        0000   393 ;|                                                                             |
                        0000   394 ;|                                                                             |
                        0000   395 ;|                                                                             |
                        0000   396 ;|                                                                             |
                        0000   397 ;|                                                                             |
                        0000   398 ;|                                                                             |
                        0000   399 ;|                                                                             |
                        0000   400 ;|                                                                             |
                        0000   401 ;|                                                                             |
                        0000   402 ;|                                                                             |
                        0000   403 ;|                                                                             |
                        0000   404 ;|                                                                             |
                        0000   405 ;|                                                                             |
                        0000   406 ;|                                                                             |
                        0000   407 ;|                                                                             |
                        0000   408 ;|                                                                             |
                        0000   409 ;|                                                                             |
                        0000   410 ;|                                                                             |
                        0000   411 ;|                                                                             |
                        0000   412 ;|                                                                             |
                        0000   413 ;|                                                                             |
                        0000   414 ;|                                                                             |
                        0000   415 ;|                                                                             |
                        0000   416 ;|                                                                             |
                        0000   417 ;|                                                                             |
                        0000   418 ;|                                                                             |
                        0000   419 ;|                                                                             |
                        0000   420 ;|                                                                             |
                        0000   421 ;|                                                                             |
                        0000   422 ;|                                                                             |
                        0000   423 ;|                                                                             |
```



```
0000   424 ;|                                                                                  |
0000   425 ;+----------------------------------------------------------------------------------+
```

```
                        0000    427          .SBTTL  "       L0011 & L0016 GATE ARRAY LOCATION"
                        0000    428 ;+-----------------------------------------------------------------------------+
                        0000    429 ;|                                                                             |
                        0000    430 ;|                                                                             |
                        0000    431 ;|                                                                             |
                        0000    432 ;|                                                                             |
                        0000    433 ;|                                                                             |
                        0000    434 ;|                                                                             |
                        0000    435 ;|                                                                             |
                        0000    436 ;|                                                                             |
                        0000    437 ;|                                                                             |
                        0000    438 ;|                                                                             |
                        0000    439 ;|                                                                             |
                        0000    440 ;|                                                                             |
                        0000    441 ;|                                                                             |
                        0000    442 ;|                                                                             |
                        0000    443 ;|                                                                             |
                        0000    444 ;|                                                                             |
                        0000    445 ;|                                                                             |
                        0000    446 ;|                                                                             |
                        0000    447 ;|                                                                             |
                        0000    448 ;|                                                                             |
                        0000    449 ;|                                                                             |
                        0000    450 ;|                                                                             |
                        0000    451 ;|                                                                             |
                        0000    452 ;|                                                                             |
                        0000    453 ;|                                                                             |
                        0000    454 ;|                                                                             |
                        0000    455 ;|                                                                             |
                        0000    456 ;|                                                                             |
                        0000    457 ;|                                                                             |
                        0000    458 ;|                                                                             |
                        0000    459 ;|                                                                             |
                        0000    460 ;|                                                                             |
                        0000    461 ;|                                                                             |
                        0000    462 ;|                                                                             |
                        0000    463 ;|                                                                             |
                        0000    464 ;|                                                                             |
                        0000    465 ;|                                                                             |
                        0000    466 ;|                                                                             |
                        0000    467 ;|                                                                             |
                        0000    468 ;|                                                                             |
                        0000    469 ;|                                                                             |
                        0000    470 ;|                                                                             |
                        0000    471 ;|                                                                             |
                        0000    472 ;|                                                                             |
                        0000    473 ;|                                                                             |
                        0000    474 ;|                                                                             |
                        0000    475 ;|                                                                             |
                        0000    476 ;|                                                                             |
                        0000    477 ;|                                                                             |
                        0000    478 ;|                                                                             |
                        0000    479 ;|                                                                             |
                        0000    480 ;|                                                                             |
                        0000    481 ;|                                                                             |
```



```
0000   482 ;|                                                                                  |
0000   483 ;+----------------------------------------------------------------------------------+
```

```
                     0000   485          .SBTTL  ""
                     0000   486          .LIST   MC
                     0000   487          .LIBRARY        /UCOD$0:[VAX750]COMETMAC/
                     0000   488          .LIST   ME
                     0000   489          .NLIST  CND,MC
                     0000                .SBTTL  "EQUATED SYMBOLS
                     0000        ;
                     0000        ; RDM REGISTER DEFINITIONS:
                     0000        ;
                     0000        ; NOTE: THESE REGITER DEFINITIONS ARE OFFSET BY THE LOAD/START ADDRESS OF
                     0000        ;       THE PROGRAM. FOR THE SIMULATOR VERSION THIS IS 400(X) AND FOR THE
                     0000        ;       REAL VERSION IT IS 8400(X).
                     0000        ;
00007400             0000                DCS_DATA_REG     =       ^XF800 + HEAD
00007401             0000                DCS_ADDR_REG_WO  =       ^XF801 + HEAD
00007402             0000                DCS_CTRL_REG     =       ^XF802 + HEAD
00007403             0000                DCS_CTRL_FILE    =       ^XF803 + HEAD
00007404             0000                ADDR_MATCH_LO    =       ^XF804 + HEAD
00007405             0000                ADDR_MATCH_HI    =       ^XF805 + HEAD
00007406             0000                STATUS_REG       =       ^XF806 + HEAD
                     0000        ;
00007410             0000                A_REG_BYTE_0     =       ^XF810 + HEAD
00007411             0000                A_REG_BYTE_1     =       ^XF811 + HEAD
00007412             0000                A_REG_BYTE_2     =       ^XF812 + HEAD
00007413             0000                A_REG_BYTE_3     =       ^XF813 + HEAD
00007414             0000                MD_REG_BYTE_0    =       ^XF814 + HEAD
00007415             0000                MD_REG_BYTE_1    =       ^XF815 + HEAD
00007416             0000                MD_REG_BYTE_2    =       ^XF816 + HEAD
00007417             0000                MD_REG_BYTE_3    =       ^XF817 + HEAD
00007418             0000                MH_REG_BYTE_0    =       ^XF818 + HEAD
00007419             0000                MH_REG_BYTE_1    =       ^XF819 + HEAD
0000741A             0000                MH_REG_BYTE_2    =       ^XF81A + HEAD
0000741B             0000                MH_REG_BYTE_3    =       ^XF81B + HEAD
0000741C             0000                CMI_CTRL_REG     =       ^XF81C + HEAD
0000741D             0000                MAINT_REG        =       ^XF81D + HEAD
0000741E             0000                RD_CTRL_REG      =       ^XF81E + HEAD
                     0000        ;
00007420             0000                FRONT_PNL_1      =       ^XF820 + HEAD
00007421             0000                FRONT_PNL_2      =       ^XF821 + HEAD
00007422             0000                CS_ADD_TRAP_LOW  =       ^XF822 + HEAD
00007423             0000                CS_ADD_TRAP_HGH  =       ^XF823 + HEAD
00007424             0000                CS_ADD_BUFF_LOW  =       ^XF824 + HEAD
00007425             0000                CS_ADD_BUFF_HGH  =       ^XF825 + HEAD
00007426             0000                BUFF_ADDR_LOW    =       ^XF826 + HEAD
00007427             0000                DCS_ADDR_REG_RO  =       ^XF827 + HEAD
                     0000        ;
0000742C             0000                WD_REG_BYTE_0    =       ^XF82C + HEAD
0000742D             0000                WD_REG_BYTE_1    =       ^XF82D + HEAD
0000742E             0000                WD_REG_BYTE_2    =       ^XF82E + HEAD
0000742F             0000                WD_REG_BYTE_3    =       ^XF82F + HEAD
00007430             0000                WH_REG_BYTE_0    =       ^XF830 + HEAD
00007431             0000                WH_REG_BYTE_1    =       ^XF831 + HEAD
00007432             0000                WH_REG_BYTE_2    =       ^XF832 + HEAD
00007433             0000                WH_REG_BYTE_3    =       ^XF833 + HEAD
                     0000        ;
```

```
                    0000        ; RDM REGISTER NAMES AS DEFINED IN HARDWARE SPEC
                    0000        ;
00007400            0000            DCSDA=DCS_DATA_REG              ;Diagnostic control store data register
00007401            0000            DCSAD=DCS_ADDR_REG_WO           ;Diagnostic control store address reg
00007402            0000            DCSCR=DCS_CTRL_REG              ;Diagnostic control store control reg
00007403            0000            DCSCF=DCS_CTRL_FILE             ;Diagnostic control store control file
00007404            0000            CSAMR=ADDR_MATCH_LO             ;Control store address match register
00007406            0000            STATUS=STATUS_REG               ;Status register
00007410            0000            MAREG=A_REG_BYTE_0              ;Memory address register
00007414            0000            MDREG=MD_REG_BYTE_0             ;Memory data register
00007418            0000            MHREG=MH_REG_BYTE_0             ;Memory holding register
0000741C            0000            CMICTL=CMI_CTRL_REG             ;CMI control
0000741D            0000            MAIN32=MAINT_REG                ;32 Bit path maintenance control
0000741E            0000            RDCTRL=RD_CTRL_REG              ;Remote diagnosis control
00007420            0000            FRONT1=FRONT_PNL_1              ;Front panel readback 1
00007421            0000            FRONT2=FRONT_PNL_2              ;Front panel readback 2
00007422            0000            CSTRP=CS_ADD_TRAP_LOW           ;Control store address trap register
00007424            0000            CSBUF=CS_ADD_BUFF_LOW           ;Control store address buffer
00007426            0000            BADD1=BUFF_ADDR_LOW             ;Control store address trace readback
00007427            0000            CLKDCS=DCS_ADDR_REG_RO          ;Clock control & DCS address register
0000742C            0000            WDREG=WD_REG_BYTE_0             ;WBUS data register
00007430            0000            WHREG=WH_REG_BYTE_0             ;WBUS holding register
                    0000        ;
                    0000        ; RDM REGISTER BIT DEFINITIONS:
                    0000        ;
                    0000        ; DCS CONTROL REGISTER
00000001            0000            HALT_ON_MATCH    =      1
00000002            0000            CLEAR_CTRL_FILE  =      2
00000004            0000            TRACE_ENABL      =      4
00000008            0000            PAR_CHK_ENABL    =      8
00000010            0000            PAR_STOP_ENABL   =      ^X10
00000020            0000            CLK_CTRL_0       =      ^X20    ; ACTIVE LOW
00000040            0000            CLK_CTRL_1       =      ^X40    ; ACTIVE LOW
00000080            0000            MICRO_ADDR_INH   =      ^X80
                    0000            ;
                    0000            ; FOLLOWING IS A FUNCTION TABLE OF THE CLOCK CONTROL BITS
                    0000            ;
                    0000                ;       CLK_CTRL 1,0    FUNCTION
                    0000                ;           11            HALT
                    0000                ;           01            SINGLE MICRO INSTRUCTION
                    0000                ;           10            SINGLE TICK
                    0000                ;           00            RUN
                    0000        ;
                    0000        ; DCS CONTROL FILE      ALL OF THESE BITS ARE ACTIVE LOW
                    0000        ;
00000001            0000            V_BUS_STROBE     =      1
00000002            0000            DCS_ADDR_CLEAR   =      2
00000004            0000            STOP_CLOCK       =      4
00000008            0000            EN_TRACE_REG     =      8
00000010            0000            H_FROM_WBUS      =      ^X10
00000020            0000            WBUS_FROM_D      =      ^X20
00000040            0000            STROBE_CMI       =      ^X40
00000080            0000            SA_CLOCK         =      ^X80
                    0000        ;
                    0000        ; CS ADDRESS MATCH REGISTER (HIGH)
```

```
                    0000        ;
00000040            0000                VBUS_SERIAL_IN  =       ^X40    ; ACTIVE LOW
00000080            0000                VBUS_CLOCK      =       ^X80
                    0000        ;
                    0000        ; STATUS REGISTER
                    0000        ;
00000001            0000                VBUS_SERIAL_OUT =       1
00000002            0000                TRAP_OFF        =       2
00000008            0000                CMI_STATUS_0    =       8
00000010            0000                CMI_STATUS_1    =       ^X10
                    0000                ;
                    0000                ; THE CMI STATUS BITS ARE ENCODED AS FOLLOWS:
                    0000                ;
                    0000                ;       00 = NO RESPONSE
                    0000                ;       01 = UNCORRECTABLE ERROR
                    0000                ;       10 = CORRECTABLE ERROR
                    0000                ;       11 = NO ERRORS
00000020            0000                CMI_COMPLETE    =       ^X20
                    0000        ;
00000080            0000                CLOCK_STOPED    =       ^X80
                    0000        ;
                    0000        ; CMI CONTROL REGISTER
                    0000        ;
00000001            0000                CMI_CTRL_CLEAR  =       1
00000008            0000                CMI_GO          =       8
00000020            0000                CMI_WRITE       =       ^X20
00000040            0000                CMI_READ        =       ^X40
00000080            0000                SA_START_STOP   =       ^X80
                    0000        ;
                    0000        ; 32 BIT MAINTENANCE CONTROL REGISTER
                    0000        ;
00000001            0000                MAINT_TRAP_OFF  =       1
00000002            0000                MAINT_STB_INH   =       2
00000004            0000                MAINT_DCS_ENABL =       4
00000008            0000                MAINT_WB_TO_WH  =       8
00000010            0000                MAINT_WD_TO_WB  =       ^X10
00000020            0000                MAINT_CMI_TO_MH =       ^X20
00000040            0000                MAINT_MD_TO_CMI =       ^X40
00000080            0000                MAINT_A_TO_CMI  =       ^X80
                    0000        ;
                    0000        ; RD CONTROL REGISTER
                    0000        ;
00000008            0000                MASTER_HALT_EN  =       8
00000010            0000                VBUS_LOAD       =       ^X10
00000020            0000                TRAP_HALT_EN    =       ^X20
00000040            0000                DC_LOW          =       ^X40
00000080            0000                AC_LOW          =       ^X80
                    0000        ;
                    0000        ; FRONT PANEL REGISTER 1
                    0000        ;
00000001            0000                FP_BOOT_0       =       1
00000002            0000                FP_BOOT_1       =       2
00000004            0000                FP_START_0      =       4
00000008            0000                FP_START_1      =       8
00000010            0000                CPU_RUN         =       ^X10
```

```
00000020  0000                PARITY_ERROR    =       ^X20
          0000        ;
          0000        ; FRONT PANEL REGISTER 2
          0000        ;
00000001  0000                REMOTE          =       1
00000002  0000                FAULT           =       2
00000004  0000                TEST            =       4
          0000        ;
          0000        ;
          0000        ;
00000040  0000                PB_INIT         =       ^X40    ; ACTIVE LOW
00000080  0000                FRONT_PNL_LOCK  =       ^X80
          0000        ;
          0000        ; DCS ADDRESS REGISTER READ ONLY
          0000        ;
00000040  0000                CLK_CTRL_0_RO   =       ^X40    ; ACTIVE HIGH
00000080  0000                CLK_CTRL_1_RO   =       ^X80    ; ACTIVE HIGH
          0000        ;
          0000        ; MISCELLANEOUS EQUATES:
          0000        ;
00000001  0000                BYTE            =       1
00000002  0000                WORD            =       2
00000004  0000                LONG            =       4
0000000A  0000                MICROWORD       =       10
00002C5E  0000                MONITOR_SIZE    =       ^X2C5E
00000400  0000                STACK_SIZE      =       1024
000007A2  0000                M$TEST_SIZE     =       14336 - STACK_SIZE - MONITOR_SIZE
          0000                                              ; MAXIMUM SIZE OF A TEST OVERLAY
000003E2  0000                M$TEST_SIZE_D   = M$TEST_SIZE - 960
          0000                                              ; MAXIMUM SIZE OF TEST OVERLAY WITH
          0000                                              ; 'DEBUG' ENABLED IN
00000001  0000                B               =       BYTE
00000002  0000                W               =       WORD
00000004  0000                L               =       LONG
0000000A  0000                M               =       MICROWORD
00000001  0000                HIGH            =       1
00000000  0000                LOW             =       0
00000000  0000                ONERROR         =       0
00000001  0000                NOERROR         =       1
00000002  0000                ONEQUAL         =       2
00000003  0000                NOTEQUAL        =       3
00000010  0000                MAX_ARGUMENTS   =       16      ; BYTE LENGTH OF LARGEST PSEUDO OP
00000036  0000                DCS_SCRATCH_ADR =       54      ; DCS SCRATCH ADDRESS START
000032FC  0000                ERROR_LOG_ADR   =       ^XB6FC + HEAD
          0000                                              ; START ADDRESS OF MEMORY ERROR LOG
          0000    491   .NLIST  ME
          0000    492   .LIST   MC
          0000    493   .SBTTL  ""
          0000    494   .NLIST  MC      ; SHUT-OFF LISTING FOR FIRST 'NEWTEST' CALL
```

```
0037          .SBTTL  "TEST   001     Check for Patch Control Store (PCS)/Initialize PCS
0037        3 ;*******************************************************************************
0037        4 ;++
0037        5 ;
0037        6 ; FUNTIONAL DESCRIPTION:
0037        7 ;
0037        8 ;       This test will check for the presence of the PCS module (L0008) and
0037        9 ;       if present then initialize all of the patch bits to 1 (non-asserted).
0037       10 ;
0037       11 ; TEST ALGORITHM:
0037       12 ;
0037       13 ;         1.  Initialize the CPU
0037       14 ;         2.  Execute the microcode to:
0037       15 ;                a.  Disable Cache
0037       16 ;                b.  Enable the CMI
0037       17 ;         3.  Set error loop address
0037       18 ;         4.  Let RDM know that we expect a micro trap
0037       19 ;         5.  Execute the microcode to:
0037       20 ;                a.  Load VA with PCS Base address
0037       21 ;                b.  Write a test pattern to VA
0037       22 ;                c.  Give time for write to complete
0037       23 ;                d.  Read the location by the VA
0037       24 ;                e.  Pass read data back to RDM
0037       25 ;         6.  Check for micro trap.  If there is a micro trap, PCS (L0008) is
0037       26 ;             not installed on the system, then do the following:
0037       27 ;                a.  Execute the microcode to:
0037       28 ;                       i.  Point to Bus Error Register
0037       29 ;                      ii.  Pass the Bus Error Register back to the RDM
0037       30 ;                b.  Check to see if NXM bit is set.
0037       31 ;                c.  If NXM bit was not set then call an error, got a bit set
0037       32 ;                    that shouldn't have been.
0037       33 ;                d.  Skip to next test
0037       34 ;          7.  Check to see that read data matches the write pattern.
0037       35 ;          8.  Call an error if read data and write pattern did not match.
0037       36 ;          9.  Set error loop address
0037       37 ;         10.  Set-up DCS at starting micro address
0037       38 ;         11.  Insure that the Clocks are turned off
0037       39 ;         12.  Load match register with 1801(hex)
0037       40 ;         13.  Clear master halt enable
0037       41 ;         14.  Set bit to watch for micro traps
0037       42 ;         15.  Start the CPU clocks
0037       43 ;         16.  Perform a Watchdog loop while executing the microcode to:
0037       44 ;              (if loop count expires then force an error, shouldn't happen)
0037       45 ;                 a.  Load VA with address of Enable Register
0037       46 ;                 b.  Enable the writing of the patch bits
0037       47 ;                 c.  Give time for the write to complete
0037       48 ;                 d.  Load the VA with the patch bit starting address
0037       49 ;                 e.  Jump to DCS address 0
0037       50 ;                 f.  Pass the current VA to the RDM
0037       51 ;                 g.  Write the patch bit to a 1 (disable the patch bit)
0037       52 ;                 h.  Give time for the write to complete
0037       53 ;                 i.  Update the VA by 4
0037       54 ;                 j.  Check to see if VA is pointing to patch bit last address
0037       55 ;                     + 4.
0037       56 ;                        i.  If VA is equal to F08000 then next address will
```

```
0037     57 ;                              be equal to 1801 and a micro match will take
0037     58 ;                              place, ELSE
0037     59 ;                          ii. If VA is not equal to F08000 then next address
0037     60 ;                              will be equal to 1800 and microcode execute
0037     61 ;                              procedes as normal.
0037     62 ;                  k. Jump to DCS address 0, (go to step 16-f)
0037     63 ;          17. Check WHREG to insure that the last patch bit address was reached.
0037     64 ;          18. If WHREG not equal to last patch bit address then call an error.
0037     65 ;          19. Execute the microcode to:
0037     66 ;                  a. Load VA with PCS/WCS or Patch bit write enable register
0037     67 ;                  b. Initialize the register with zero
0037     68 ;                  c. Return back to the RDM
0037     69 ;          20. Skip to next test
0037     70 ;
0037     71 ; TEST PATTERNS:
0037     72 ;
0037     73 ;          The following test pattern is used when writing PCS to see if it is
0037     74 ;          present:
0037     75 ;
0037     76 ;                  000AAAAA
0037     77 ;
0037     78 ; LOGIC DESCRIPTION:
0037     79 ;
0037     80 ;          With the PCS module installed this test will check the CMI to PCS
0037     81 ;          interface logic to insure that, 1)  it will respond to its physical
0037     82 ;          address, and, 2) the data line tranceivers are working by writing
0037     83 ;          then reading a test pattern.
0037     84 ;
0037     85 ; ASSUMPTIONS:
0037     86 ;
0037     87 ;          None
0037     88 ;
0037     89 ; ERROR DESCRIPTION:
0037     90 ;
0037     91 ;          First IFERROR:
0037     92 ;
0037     93 ;                  Expected Read Data
0037     94 ;                  Received Read Data
0037     95 ;
0037     96 ;          Second IFERROR:
0037     97 ;
0037     98 ;                  WHREG     <--\  These are NOT significant as the IFERROR is
0037     99 ;                  0F000000  <--/  used only to FORCE an error.
0037    100 ;
0037    101 ;          Third IFERROR:
0037    102 ;
0037    103 ;                  Expected Patch Bit Last Address
0037    104 ;                  Received Patch Bit Last Address
0037    105 ;
0037    106 ;          Fourth IFERROR:
0037    107 ;
0037    108 ;                  Expected Bus Error Register
0037    109 ;                  Received Bus Error Register
0037    110 ;
0037    111 ;--
```

```
0037   ;*****************************************************************************
0037
0037           INITIALIZE                          ; Initialize the CPU
0039   ;
0039   ; Disable Cache and Turn the CMI on
0039   ;
0039           FETCH           <^X17>              ; start dcs program
003E           BRSTCLK         <^X1C>,INH          ; end dcs program
0042   ;
0042   ; Check to see if there is a PCS on this system
0042   ;
0042           ERRLOOP                             ; loop on error from here
0044           EXPUTRAP                            ; Expect micro trap
0046           FETCH           <^X0F>              ; start dcs program
004B           BRSTCLK         <^X16>              ; end dcs program
004F           CMPREGMSK       CSTRP,7$,,8$,,W,NE
005B                                               ; Was there a machine check
005B           SKIP            150$,ONERROR        ; Go handle machine check
0062           CMPREGMSK       WHREG,9$,,12$,,L; Check the Read Data
006E           IFERROR         ,,<CCS>             ; Call error if data didn't match
0074   ;
0074   ; This routine will startup the micro code which will initialize the Patch
0074   ; bits to a One.
0074   ;
0074           ERRLOOP                             ; loop on error from here
0076           FETCH           <^X08>              ; Set-up to start DCS
007B           LOADREG         DCSCR,1$,,B         ; Insure that the clocks are off
0083           LOADREG         CSAMR,2$,,W         ; Load the match address
008B           LOADREG         RDCTRL,3$,,B        ; Clear master halt enable
0093           LOADREG         RDCTRL,4$,,B        ; Watch for micro-traps
009B           LOADREG         DCSCR,5$,,B         ; Start the clocks
00A3           LOOP            I,1,4096            ; Set up watch-dog loop counter
00AA           CMPREGMSK       CLKDCS,3$,,6$,,B,
00B6                                               ; Test to see if clock has stopped
00B6           SKIP            100$,ONERROR        ; Exit loop if clock stops (normal)
00BD           ENDLOOP         I                   ; end of watch-dog loop
00C0           CMPREG          WHREG,11$,,L,       ; Force an error, program shouldn't
00C9           IFERROR         ,,<CCS>             ; get here from the loop
00CF           FETCH           <^X21>              ; start dcs program to init register
00D4           BRSTCLK         <^X25>              ; end dcs program
00D8           SKIP
00DF   100$:
00DF           CMPREGMSK       WHREG,14$,,13$,,L
00EB                                               ; check to make sure correct ending
00EB                                               ; address
00EB           IFERROR         ,,<CCS>             ; Call error if not correct address
00F1           SKIP
00F8   ;
00F8   ; This is the machine check handler routine.  It makes sure that the Non
00F8   ; Existent memory bit is set which should be the case.
00F8   ;
00F8   150$:   FETCH           <^X1D>              ; Start DCS program
00FD           BRSTCLK         <^X20>,INH          ; End DCS program
0101           CMPREGMSK       WHREG,10$,,11$,,L
010D                                               ; Check to see if Non-Existent memory
```

```
                                010D          IFERROR        ,,<CCS>          ; A Bit is set that shouldn't be
                                0113          SKIP
                                011A
                                011A
                                011A  BEGIN_TEST_DATA
                                011A
                                011A      ;-----------------------------------------------------------------------
                                011A
                          60    011A  1$:     .BYTE   ^X60                     ; Data to shut the clocks off
                        1801    011B  2$:     .WORD   ^X1801                   ; Match Address
                          00    011D  3$:     .BYTE   ^X00                     ; Data to clear master halt enable
                          28    011E  4$:     .BYTE   ^X28                     ; Data to watch for u-traps
                          11    011F  5$:     .BYTE   ^X11                     ; Run clock bits
                          C0    0120  6$:     .BYTE   ^XC0                     ; Mask for clock stop bits
                        0028    0121  7$:     .WORD   ^X0028                   ; Machine Check miro trap
                        3FFF    0123  8$:     .WORD   ^X3FFF                   ; Mask for cmpregmsk pseudo op
                    000AAAAA    0125  9$:     .LONG   ^X000AAAAA               ; Expected data if PCS present
                    0A000000    0129  10$:    .LONG   ^X0A000000               ; Expected Bus Error bits
                    0F000000    012D  11$:    .LONG   ^X0F000000               ; Mask for Bus Error Reg bits
                    000FFFFF    0131  12$:    .LONG   ^X000FFFFF               ; Mask for Received PCS Data
                    00FFFFFF    0135  13$:    .LONG   ^X00FFFFFF               ; Mask for VA compare
                    00F07FFC    0139  14$:    .LONG   ^X00F07FFC               ; Last patch bit address
                                013D
                                013D  BEGIN_CPU_DATA
                                0002
                                0002  DCS_DATA
                                0001  ;
                                0001  ; The following routine will set all of the patch bits to a 1.
                                0001  ;
                                0001
U 00, 7700,C800,0364,0300,0470,1802  0001  ;% 00   NOP,NEXT/1802                   ; Get the CPU up to speed
U 01, 6700,881B,5924,0300,0470,1802  000C  ;% 01   RDM_WB,WB_M[VA]                 ; Save the VA in the RDM
U 02, 7700,8800,5BE4,0304,5C80,1803  0017  ;% 02   WRITE.PHY R[TEMP1]              ; Write the Patch bit
U 03, 7700,4800,0364,0300,0470,9804  0022  ;% 03   NOP,CLOCK.EXT                   ; Wait for write to finish
U 04, 7700,C800,0364,0300,0470,9805  002D  ;% 04   NOP,CLOCK.EXT                   ; ...
U 05, 7700,C800,0364,0300,4470,1806  0038  ;% 05   VA_VA+4                         ; Update the VA for next patch bit
U 06, 7700,081B,0034,A310,0470,1800  0043  ;% 06   WB_M[VA].XOR.R[TEMP4],BUT/WX.EQ.0,NEXT/1800
                                004E                                             ; check to see if finished
U 07, 7500,C800,0364,0300,0470,1808  004E  ;% 07   NOP,DCS.ADDR_0                  ; Continue with patch bit initialize
U 08, 7700,4800,0364,0300,0470,1809  0059  ;% 08   NOP
U 09, 7700,0800,5BE4,031C,4A70,180A  0064  ;% 09   VA_R[TEMP7]                     ; Load VA with register address to
                                006F                                             ; enable the writing of patch bits
U 0A, 7700,0800,5BE4,0304,5C80,180B  006F  ;% 0A   WRITE.PHY R[TEMP1]              ; Enable to write patch bits
U 0B, 7700,C800,0364,0300,0470,980C  007A  ;% 0B   NOP,CLOCK.EXT                   ; Wait for write to finish
U 0C, 7700,4800,0364,0300,0470,980D  0085  ;% 0C   NOP,CLOCK.EXT                   ; ...
U 0D, 7700,C800,5BE4,0314,4A70,180E  0090  ;% 0D   VA_R[TEMP5]                     ; Load VA with Patch Bit address
U 0E, 7500,4800,0364,0300,0470,180F  009B  ;% 0E   NOP,DCS.ADDR_0                  ; Go init the patch bits
                                00A6
                                00A6  ;
                                00A6  ; The following routine will first write a test pattern to the first PCS
                                00A6  ; then read the test data.  The data will be sent back to the RDM and if no
                                00A6  ; PCS is present then the data will no be valid.
                                00A6  ;
                                00A6
U 0F, 7700,C800,0364,0300,0470,1810  00A6  ;% 0F   NOP                             ; Get the CPU up to speed
```

```
U 10. 7700,0800,5BE4,0310,4A70,1811  00B1   268 ;% 10   VA_R[TEMP4]                    ; Load VA with PCS Base Address
U 11. 7700,C800,5BE4,0318,5C80,1812  00BC   272 ;% 11   WRITE.PHY R[TEMP6]            ; Write the Test pattern
U 12. 7700,4800,0364,0300,0470,9813  00C7   276 ;% 12   NOP,CLOCK.EXT                ; Give write time to complete
U 13. 7700,C800,0364,0300,0470,9814  00D2   280 ;% 13   NOP,CLOCK.EXT                ; ...
U 14. 7700,4800,0364,0300,0400,1815  00DD   284 ;% 14   READ.PHY                     ; Now Read the Test location
U 15. 6700,8812,5924,0300,0470,1816  00E8   288 ;% 15   RDM_WB,WB_M[MDR]             ; Pass the read data back to the RDM
U 16. 7300,4800,0364,0300,0470,1817  00F3   292 ;% 16   NOP,STOP.CLOCK              ; Return to the RDM
                                     00FE   296
                                     00FE   297 ;
                                     00FE   298 ; The following routine will Disable Cache and Enable the CMI
                                     00FE   299 ;
                                     00FE   300
U 17. 7700,4800,0364,0300,0470,1818  00FE   301 ;% 17   NOP                          ; Get CPU up to speed
U 18. 7700,C800,5BE4,0300,6870,1819  0109   305 ;% 18   MEMSCAR_R[TEMP0]            ; Point MEMSCAR to Cache Disable reg
U 19. 7700,0800,5BE4,0304,6070,181A  0114   309 ;% 19   MEMSCR_R[TEMP1]            ; Disable Cache
U 1A. 7700,0800,5BE4,0308,6870,181B  011F   313 ;% 1A   MEMSCAR_R[TEMP2]           ; Point MEMSCAR to CMI Disable reg
U 1B. 7700,4800,5BE4,030C,6070,181C  012A   317 ;% 1B   MEMSCR_R[TEMP3]            ; Enable the CMI
U 1C. 7300,4800,0364,0300,0470,181D  0135   321 ;% 1C   NOP,STOP.CLOCK             ; Return to the RDM
                                     0140   325
                                     0140   326 ;
                                     0140   327 ; The following routine will read the Bus Error Register.
                                     0140   328 ;
                                     0140   329
U 1D. 7700,4800,0364,0300,0470,181E  0140   330 ;% 1D   NOP                          ; Get CPU up to speed
U 1E. 7700,1800,CB70,0304,E870,181F  014B   334 ;% 1E   MEMSCAR_ZLIT24[09]         ; Point MEMSCAR to Bus Error Register
U 1F. 6700,C800,0364,0300,6470,1820  0156   338 ;% 1F   RDM_WB,WCTRL/MEMSCR        ; Read Bus Error Register
U 20. 7300,4800,0364,0300,0470,1821  0161   342 ;% 20   NOP,STOP.CLOCK             ; Return to the RDM
                                     016C   346
                                     016C   347 ;
                                     016C   348 ; The following routine will reset the PCS/WCS or Patch bit write enable
                                     016C   349 ; register back to zero.
                                     016C   350 ;
                                     016C   351
U 21. 7700,4800,0364,0300,0470,1822  016C   352 ;% 21   NOP                          ; Get CPU up to speed
U 22. 7700,8800,5BE4,031C,4A70,1823  0177   356 ;% 22   VA_R[TEMP7]                 ; load VA with register address
U 23. 7700,C800,5BE4,03D8,5C80,1824  0182   360 ;% 23   WRITE.PHY R[ZERO]          ; clear the register
U 24. 7700,4800,0364,0300,0470,9825  018D   364 ;% 24   NOP,CLOCK.EXT              ; give write time to complete
U 25. 7300,C800,0364,0300,0470,1826  0198   368 ;% 25   STOP.CLOCK                 ; Return to the RDM
                                     01A3   372
                                     01A3   373 RTEMP_DATA
                                     0001   374
                          06000000   0001   375         .LONG   ^X06000000       ; Cache Disable Register Address
                          00000001   0005   376         .LONG   ^X00000001       ; Cache Disbble Register Data
                          0E000000   0009   377         .LONG   ^X0E000000       ; CMI Disable Register Address
                          00000000   000D   378         .LONG   ^X00000000       ; CMI Disable Register Data
                          00F08000   0011   379         .LONG   ^X00F08000       ; Starting Address of PCS
                          00F00000   0015   380         .LONG   ^X00F00000       ; Starting Address for Patch Bits
                          000AAAAA   0019   381         .LONG   ^X000AAAAA       ; Test pattern used to check for PCS
                                     001D   382                                  ; present - Note that Bit 20 is 0 which
                                     001D   383                                  ; will force PCS Present to be Not True
                          00F0C000   001D   384         .LONG   ^X00F0C000       ; WCS/PCS or Patch Bit write enable reg
                                     0021   385
                                     0021   386 END_CPU_DATA
                                     013D   387
                                     013D   388 END_TEST_DATA
```

```
                    013D
                    013D        ;-----------------------------------------------------------------------------
                    013D    389
                    013D    390 ENDTEST
```

```
                        002D        .SBTTL  "TEST   002     Test Control Store Address Bus Integrity
                        002D    392 ;*****************************************************************************
                        002D    393 ;++
                        002D    394 ;
                        002D    395 ; FUNCTIONAL DESCRIPTION:
                        002D    396 ;
                        002D    397 ;       THIS TEST WILL VERIFY THAT THE CONTROL STORE ADDRESS BUS IS NOT STUCK
                        002D    398 ;       AT ZERO OR ONE.
                        002D    399 ;
                        002D    400 ;
                        002D    401 ; TEST ALGORITHM:
                        002D    402 ;
                        002D    403 ;       1.  CREATE A TEST LOOP OF 5
                        002D    404 ;       2.  LOAD MICRO ADDR INH AND PARCHK INTO THE DCSCR (RDM)
                        002D    405 ;       3.  LOAD A TEST PATTERN INTO THE CSAMR (RDM)
                        002D    406 ;       4.  TEST THE CSBUF REGISTER (RDM) FOR TEST PATTERN
                        002D    407 ;       5.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST PATTERNS
                        002D    408 ;
                        002D    409 ;
                        002D    410 ; TEST PATTERNS:
                        002D    411 ;
                        002D    412 ;       ITERATION       TEST PATTERN
                        002D    413 ;           1               2AAA
                        002D    414 ;           2               1555
                        002D    415 ;           3               3333
                        002D    416 ;           4               0F0F
                        002D    417 ;           5               00FF
                        002D    418 ;
                        002D    419 ;
                        002D    420 ; LOGIC DESCRIPTION:
                        002D    421 ;
                        002D    422 ;       THIS TEST WILL ENSURE THAT THE CONTROL STORE ADDRESS BUS IS NOT STUCK
                        002D    423 ;       AT ONE OR ZERO.  TEST PATTERNS ARE DRIVEN FROM THE RDM CSAMR REGISTER
                        002D    424 ;       ONTO THE CS ADDRESS BUS AND READ BACK TO THE RDM VIA THE CSBUF
                        002D    425 ;       REGISTER.  ANY ERRORS WOULD INDICATE THAT THE BUS IS STUCK OR BEING
                        002D    426 ;       DRIVEN BY ANOTHER MODULE.
                        002D    427 ;
                        002D    428 ;
                        002D    429 ; ASSUMPTIONS:
                        002D    430 ;
                        002D    431 ;       THIS TEST ASSUMES THE RDM IS OPERATIONAL.
                        002D    432 ;
                        002D    433 ;
                        002D    434 ; ERROR DESCRIPTION:
                        002D    435 ;
                        002D    436 ;       EXPECTED ADDRESS ON CS BUS
                        002D    437 ;       RECEIVED ADDRESS ON CS BUS
                        002D    438 ;       LOOP COUNT I
                        002D    439 ;
                        002D    440 ;--
                        002D    441 ;*****************************************************************************
                        002D    442
                        002D    443
                        002D    444         INITIALIZE                              ;INIT THE CPU
                        002F    445         LOOP            I,1,5                   ;CREATE A TEST LOOP OF 5
```

```
                0036   446          ERRLOOP                                 ;ERROR LOOP POINT
                0038   447          LOADREG         DCSCR,1$,,B             ;LOAD UADD INH & PARCHK
                0040   448          LOADREG         CSAMR,2$,I,W            ;LOAD TEST PATTERN INTO CSAMR
                0048   449          CMPREGMSK       CSBUF,2$,I,3$,,W,       ;TEST FOR CORRECT TEST PATTERN
                0054   450          IFERROR         ,,<<DPM><CCS>>          ;CS BUS STUCK
                005B   451          LOADREG         DCSCR,4$,,B             ;CLEAR DCSCR
                0063   452          ENDLOOP         I                       ;END TEST LOOP
                0066   453          SKIP                                    ;GO TO NEXT TEST
                006D   454
                006D   455
                006D   456 BEGIN_TEST_DATA
                006D
                006D       ;-------------------------------------------------------------------------
                006D   457
              E8 006D   458 1$:      .BYTE   ^XE8
            2AAA 006E   459 2$:      .WORD   ^X2AAA                          ;TEST PATTERN TABLE
            1555 0070   460          .WORD   ^X1555
            3333 0072   461          .WORD   ^X3333
            0F0F 0074   462          .WORD   ^X0F0F
            00FF 0076   463          .WORD   ^X00FF
            1FFF 0078   464 3$:      .WORD   ^X1FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
            0060 007A   465 4$:      .WORD   ^X60                            ;CLEAR DCSCR
                007C   466
                007C   467 END_TEST_DATA
                007C
                007C       ;-------------------------------------------------------------------------
                007C   468 ENDTEST
```

```
0018          .SBTTL  "TEST   003     Wbus Data Integrity
0018    470 ;*****************************************************************************
0018    471 ;++
0018    472 ;
0018    473 ; FUNCTIONAL DESCRIPTION:
0018    474 ;
0018    475 ;       THIS TEST WILL CHECK THAT THE WBUS IS NOT STUCK AT ONE OR ZERO.
0018    476 ;
0018    477 ;
0018    478 ; TEST ALGORITHM:
0018    479 ;
0018    480 ;       1.  CREATE A TEST LOOP OF 6
0018    481 ;       2.  GENERATE A TEST PATTERN
0018    482 ;       3.  LOAD TEST PATTERN INTO THE RDM WDREG
0018    483 ;       4.  ENABLE THE WDREG ONTO THE WBUS
0018    484 ;       5.  READ THE WBUS BACK TO THE RDM WHREG
0018    485 ;       6.  TEST THE WHREG FOR THE CORRECT TEST PATTERN
0018    486 ;       7.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING PATTERNS
0018    487 ;
0018    488 ;
0018    489 ; TEST PATTERNS:
0018    490 ;
0018    491 ;       ITERATION       TEST PATTERN
0018    492 ;           1             AAAAAAAA
0018    493 ;           2             55555555
0018    494 ;           3             33333333
0018    495 ;           4             0F0F0F0F
0018    496 ;           5             00FF00FF
0018    497 ;           6             0000FFFF
0018    498 ;
0018    499 ;
0018    500 ; LOGIC DESCRIPTION:
0018    501 ;
0018    502 ;       THIS TEST VERIFIES THE WBUS BY DRIVING TEST PATTERNS OUT OF THE RDM
0018    503 ;       WDREG AND READING THEM BACK VIA THE WHREG.  ANY FAILURES WOULD INDICATE
0018    504 ;       THE BUS IS STUCK OR ONE OF THE OTHER MODULES ON THE BUS IS TRYING TO
0018    505 ;       DRIVE IT AT THE SAME TIME.
0018    506 ;
0018    507 ;
0018    508 ; ASSUMPTIONS:
0018    509 ;
0018    510 ;       THIS TEST ASSUMES THE RDM IS OPERATIONAL.
0018    511 ;
0018    512 ;
0018    513 ; ERROR DESCRIPTION:
0018    514 ;
0018    515 ;       EXPECTED WBUS CONTENTS
0018    516 ;       RECEIVED WBUS CONTENTS
0018    517 ;       LOOP COUNT I
0018    518 ;
0018    519 ;--
0018    520 ;*****************************************************************************
0018    521
0018    522
0018    523         INITIALIZE                                      ;INIT THE CPU
```

```
                001A          LOOP          I,1,6                    ;CREATE A TEST LOOP OF 6
                0021          SA01PAT       I,1$,,                   ;GENERATE A TEST PATTERN
                0029          ERRLOOP                                ;ERROR LOOP POINT
                002B          LOADREG       WDREG,1$,,L              ;LOAD TEST PATTERN INTO WDREG
                0033          LOADREG       MAIN32,2$,,B             ;ENABLE WDREG ONTO WBUS
                003B          LOADREG       MAIN32,3$,,B             ;STROBE WHREG
                0043          LOADREG       MAIN32,2$,,B             ;...
                004B          LOADREG       MAIN32,4$,,B             ;DISABLE WDREG
                0053          CMPREG        WHREG,1$,,L,             ;TEST FOR THE CORRECT TEST PAT.
                005C          IFERROR       ,,<<MIC><UBI><FPA>>      ;BITS STUCK ON WBUS
                0064          ENDLOOP       I                        ;END TEST LOOP
                0067          SKIP                                   ;GO TO NEXT TEST
                006E
                006E
                006E  BEGIN_TEST_DATA
                006E
                006E      ;-----------------------------------------------------------------------
                006E
        00000000 006E  1$:     .LONG   ^X00000000                    ;TEST PATTERN STORAGE
              15 0072  2$:     .BYTE   ^X15                          ;ENABLE WDREG TO WBUS
              1D 0073  3$:     .BYTE   ^X1D                          ;ENABLE WHREG
              05 0074  4$:     .BYTE   ^X05                          ;DISABLE WDREG
                0075
                0075  END_TEST_DATA
                0075
                0075      ;-----------------------------------------------------------------------
                0075  ENDTEST
```

```
0013          .SBTTL  "TEST   004     Test CPU Clock
0013    548 ;*****************************************************************************
0013    549 ;++
0013    550 ;
0013    551 ; FUNCTIONAL DESCRIPTION:
0013    552 ;
0013    553 ;       THIS TEST WILL VERIFY THAT THE CPU B CLOCK CAN BE HALTED AND RUN
0013    554 ;       FROM THE RDM.
0013    555 ;
0013    556 ;
0013    557 ; TEST ALGORITHM:
0013    558 ;
0013    559 ;       1.  LOAD THE RDM DCSCR REGISTER WITH HALT THE CLOCK
0013    560 ;       2.  TEST THE RDM STATUS REGISTER TO VERIFY THAT B CLOCK IS HALTED
0013    561 ;       3.  IF NO ERROR SET DC LOW IN THE RDM RDCTRL REGISTER
0013    562 ;       4.  STROBE THE VBUS
0013    563 ;       5.  TEST THE VBUS FOR MSEQ INIT AND DISABLE HI NEXT
0013    564 ;       6.  IF NO ERROR TEST THE STATUS REGISTER FOR B CLOCK RUNNING
0013    565 ;       7.  IF NO ERROR RELEASE DC LOW AND GO TO NEXT TEST
0013    566 ;
0013    567 ;
0013    568 ; TEST PATTERNS:
0013    569 ;
0013    570 ;       NONE
0013    571 ;
0013    572 ;
0013    573 ; LOGIC DESCRIPTION:
0013    574 ;
0013    575 ;       THIS TEST CHECKS THAT THE CPU B CLOCK CAN BE SET TO THE HALT AND
0013    576 ;       RUN CONDITIONS FROM THE RDM.  ALSO MSEQ INIT AND DISABLE HI NEXT ARE
0013    577 ;       TESTED VIA THE VBUS TO SEE IF THEY ARE SET BY DC LOW.
0013    578 ;
0013    579 ;
0013    580 ; ASSUMPTIONS:
0013    581 ;
0013    582 ;       THIS TEST ASSUMES THE RDM IS OPERATIONAL.
0013    583 ;
0013    584 ;
0013    585 ; ERROR DESCRIPTION:
0013    586 ;
0013    587 ;       FIRST AND THIRD IFERROR:
0013    588 ;               EXPECTED CONTENTS OF STATUS REGISTER
0013    589 ;               RECEIVED CONTENTS OF STATUS REGISTER
0013    590 ;
0013    591 ;       SECOND IFERROR:
0013    592 ;               EXPECTED VBUS BITS
0013    593 ;               RECEIVED VBUS BITS
0013    594 ;                       (BITS <7:0> = VBUS BIT POSITION)
0013    595 ;                       (BITS <8> = BIT STATE)
0013    596 ;
0013    597 ;--
0013    598 ;*****************************************************************************
0013    599
0013    600
0013    601         INITIALIZE                                      ;INIT THE CPU
```

```
                0015   602          LOADREG         DCSCR,1$,,B                ;HALT THE CLOCK
                001D   603          CMPREGMSK       STATUS,2$,,3$,,B,          ;TEST THE CLOCK DETECTION BIT
                0029   604          IFERROR         ,<<SAC>>,                  ;BIT INCORRECT
                002F   605          LOADREG         RDCTRL,4$,,B               ;HOLD DC LOW TO RUN CLOCK
                0037   606          LOADREG         RDCTRL,7$,,B               ;STROBE THE VBUS
                003F   607          LOADREG         RDCTRL,4$,,B               ;...
                0047   608          CMPVBUS         6$                         ;TEST FOR MSEQ INIT/DISABLE HI
                004C   609    .     IFERROR         ,<<MSQ>>,<<UBI>>           ;MSEQ INIT/DISABLE HI MISSING
                0053   610          CMPREGMSK       STATUS,5$,,3$,,B,          ;TEST THE CLOCK DETECTION BIT
                005F   611          IFERROR         ,<<SAC>>,                  ;BIT INCORRECT
                0065   612          LOADREG         RDCTRL,5$,,B               ;FREE DC LOW
                006D   613          SKIP                                       ;GO TO NEXT TEST
                0074   614
                0074   615
                0074   616 BEGIN_TEST_DATA
                0074
                0074       ;-------------------------------------------------------------------------
                0074   617
          60    0074   618 1$:      .BYTE   CLK_CTRL_0!CLK_CTRL_1              ;HALT THE CLOCK
          80    0075   619 2$:      .BYTE   ^X80
          80    0076   620 3$:      .BYTE   ^X80                               ;MASK FOR CMPREGMSK PSEUDO OP
          40    0077   621 4$:      .BYTE   DC_LOW                             ;FORCE DC LOW
          00    0078   622 5$:      .BYTE   ^X00                               ;COMPARE DATA
          02    0079   623 6$:      .BYTE   ^X2
                007A   624          VBUSDATA        <^X1C>,1                   ;DISABLE HI NEXT
                007B   625          VBUSDATA        <^X27>,0                   ;MSEQ INIT ON VBUS
          50    007C   626 7$:      .BYTE   DC_LOW!VBUS_LOAD                   ;LOAD THE VBUS
                007D   627
                007D   628 END_TEST_DATA
                007D
                007D       ;-------------------------------------------------------------------------
                007D   629 ENDTEST
```

```
                        0028        .SBTTL  "TEST   005     Test CPU Clock Single Tick and Step
                        0028    631 ;*******************************************************************************
                        0028    632 ;++
                        0028    633 ;
                        0028    634 ; FUNCTIONAL DESCRIPTION:
                        0028    635 ;
                        0028    636 ;       THIS TEST CHECKS THAT THE SINGLE TICK AND SINGLE STEP CLOCK FUNCTIONS
                        0028    637 ;       ARE OPERATIONAL.
                        0028    638 ;
                        0028    639 ;
                        0028    640 ; TEST ALGORITHM:
                        0028    641 ;
                        0028    642 ;         1.  LOAD 1800 INTO THE RDM CSAMR REGISTER
                        0028    643 ;         2.  SET PARCHK, UADINH AND TRACE IN THE RDM DCSCR
                        0028    644 ;         3.  CREATE A TEST LOOP OF 2
                        0028    645 ;         4.  SINGLE TICK THE CLOCK VIA THE RDM DCSCR
                        0028    646 ;         5.  REPEAT STEPS 3 AND 4 FOR ANOTHER SINGLE TICK
                        0028    647 ;         6.  STROBE THE VBUS
                        0028    648 ;         7.  TEST THE VBUS FOR DISABLE HI NEXT
                        0028    649 ;         8.  IF NO ERROR TEST THE RDM CSTRP FOR 1800
                        0028    650 ;         9.  IF NO ERROR LOAD THE RDM CSAMR REGISTER WITH 18FF
                        0028    651 ;        10.  SET PARCHK, UADINH AND TRACE IN THE RDM DCSCR
                        0028    652 ;        11.  SINGLE STEP THE CPU CLOCK VIA THE RDM DCSCR
                        0028    653 ;        12.  TEST THE RDM CSTRP REGISTER FOR 18FF
                        0028    654 ;        13.  IF NO ERROR GO TO NEXT TEST
                        0028    655 ;
                        0028    656 ;
                        0028    657 ; TEST PATTERNS:
                        0028    658 ;
                        0028    659 ;       SINGLE TICK:
                        0028    660 ;               CS ADDRESS = 1800
                        0028    661 ;
                        0028    662 ;       SINGLE STEP:
                        0028    663 ;               CS ADDRESS = 18FF
                        0028    664 ;
                        0028    665 ;
                        0028    666 ; LOGIC DESCRIPTION:
                        0028    667 ;
                        0028    668 ;       THIS TEST WILL CHECK THAT THE CPU CLOCK CONTROL LOGIC WILL RESPOND
                        0028    669 ;       TO SINGLE TICK AND SINGLE STEP COMMANDS FROM THE RDM.  SINCE THE RDM
                        0028    670 ;       CAN'T DIRECTLY READ M CLOCK THE CSTRP REGISTER IS USED TO VERIFY THAT
                        0028    671 ;       AN M CLOCK HAS OCCURRED (CSTRP IS DRIVEN BY M CLOCK).  ALSO TESTED IS
                        0028    672 ;       THE SIGNAL DISABLE HI NEXT WHICH SHOULD BE ASSERTED AS A RESULT OF
                        0028    673 ;       UADINH.
                        0028    674 ;
                        0028    675 ;
                        0028    676 ; ASSUMPTIONS:
                        0028    677 ;
                        0028    678 ;       THIS TEST ASSUMES THE CS ADDRESS BUS IS OPERATIONAL.
                        0028    679 ;
                        0028    680 ;
                        0028    681 ; ERROR DESCRIPTION:
                        0028    682 ;
                        0028    683 ;       FIRST IFERROR:
                        0028    684 ;               EXPECTED CONTENTS OF VBUS
```

```
                        0028    685 ;                   RECEIVED CONTENTS OF VBUS
                        0028    686 ;                           (BITS <7:0> = VBUS BIT POSITION)
                        0028    687 ;                           (BITS <8> = BIT STATE)
                        0028    688 ;
                        0028    689 ;       SECOND AND THIRD IFERROR:
                        0028    690 ;                   EXPECTED CS ADDRESS
                        0028    691 ;                   RECEIVED CS ADDRESS
                        0028    692 ;
                        0028    693 ;--
                        0028    694 ;*******************************************************************************
                        0028    695
                        0028    696
                        0028    697         INITIALIZE                              ;INIT THE CPU
                        002A    698         LOADREG         CSAMR,1$,,W             ;LOAD 1800 INTO CSAMR
                        0032    699         LOADREG         DCSCR,2$,,B             ;SET PARCHK & UADINH & TRACE
                        003A    700         LOOP            I,1,2                   ;LOOP TO TICK THE CLOCK
                        0041    701         LOADREG         DCSCR,3$,,B             ;SINGLE TICK THE CLOCK
                        0049    702         LOADREG         DCSCR,2$,,B             ;STOP THE CLOCK
                        0051    703         ENDLOOP         I                       ;END CLOCK LOOP
                        0054    704         LOADREG         RDCTRL,8$,,B            ;STROBE THE VBUS FOR DISABLE HI
                        005C    705         LOADREG         RDCTRL,9$,,B            ;...
                        0064    706         LOADREG         DCSCR,5$,,B             ;HALT THE CLOCK
                        006C    707         CMPVBUS         10$                     ;TEST FOR DISABLE HI NEXT
                        0071    708         IFERROR         ,<<MSQ>>,               ;NOT PRESENT ON VBUS
                        0077    709         CMPREGMSK       CSTRP,1$,,4$,,W,        ;TEST CSTRP FOR 1800
                        0083    710         IFERROR         ,<SAC>,                 ;NO CLOCK
                        0089    711         INITIALIZE                              ;CLEAR ANY PARITY ERRORS
                        008B    712         LOADREG         CSAMR,7$,,W             ;LOAD CSAMR WITH ADDRESS 18FF
                        0093    713         LOADREG         DCSCR,2$,,B             ;SET PARCHK & UADINH & TRACE
                        009B    714         LOADREG         DCSCR,6$,,B             ;SINGLE STEP THE CLOCK
                        00A3    715         LOADREG         DCSCR,5$,,B             ;HALT THE CLOCK
                        00AB    716         CMPREGMSK       CSTRP,7$,,4$,,W,        ;TEST CSTRP FOR 18FF
                        00B7    717         IFERROR         ,<<SAC>>,               ;NO CLOCK
                        00BD    718         INITIALIZE                              ;CLEAR ANY PARITY ERRORS
                        00BF    719         SKIP                                    ;GO TO NEXT TEST
                        00C6    720
                        00C6    721
                        00C6    722 BEGIN_TEST_DATA
                        00C6
                        00C6        ;-----------------------------------------------------------------------
                        00C6    723
1800                    00C6    724 1$:     .WORD   ^X1800                          ;ADDRESS FOR CSAMR REGISTER
  EE                    00C8    725 2$:     .BYTE   ^XEE                            ;HALT THE CLOCK
  CC                    00C9    726 3$:     .BYTE   ^XCC                            ;SINGLE TICK THE CPU CLOCK
3FFF                    00CA    727 4$:     .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
  60                    00CC    728 5$:     .BYTE   ^X60                            ;HALT THE CLOCK
  AC                    00CD    729 6$:     .BYTE   ^XAC                            ;SINGLE STEP THE CPU CLOCK
18FF                    00CE    730 7$:     .WORD   ^X18FF                          ;ADDRESS FOR SINGLE STEP
  10                    00D0    731 8$:     .BYTE   ^X10                            ;LOAD VBUS
  00                    00D1    732 9$:     .BYTE   ^X00                            ;CLEAR RDCTRL REGISTER
  01                    00D2    733 10$:    .BYTE   ^X01
                        00D3    734         VBUSDATA        <^X1C>,1                ;DISABLE HI NEXT ON VBUS
                        00D4    735
                        00D4    736 END_TEST_DATA
                        00D4
```

```
00D4        ;-----------------------------------------------------------------------
00D4    737 ENDTEST
```

```
                        .SBTTL  "TEST   006     Control Store Bus Integrity
0020    739 ;*****************************************************************************
0020    740 ;++
0020    741 ;
0020    742 ; FUNCTIONAL DESCRIPTION
0020    743 ;
0020    744 ;       This test isolates single bits stuck high or stuck low and can detect
0020    745 ;       two bits stuck high or low or tied to each other on the CS bus.
0020    746 ;
0020    747 ; TEST ALGORITHM
0020    748 ;
0020    749 ;       This test is implemented in five subtests. Subtests 1-4 isolate single
0020    750 ;       bit errors, while subtest 5 will detect double bit errors.
0020    751 ;
0020    752 ;       Subtest 1
0020    753 ;
0020    754 ;       This subtest checks for one bit in the odd parity field stuck at one.
0020    755 ;
0020    756 ;        1. Place zeros onto CPU control store lines.
0020    757 ;        2. Examine the front panel register for a parity error.
0020    758 ;        3. If there is a parity error, skip to subtest 2.
0020    759 ;        4. Prepare a loop of 40 iterations. (One iteration per bit)
0020    760 ;        5. Set a bit in the odd parity field, indexed by the loop count.
0020    761 ;        6. Is there a parity error?
0020    762 ;        7. If not, print out the loop count to identify the bad bit.
0020    763 ;        8. Otherwise, continue looping.
0020    764 ;
0020    765 ;       Subtest 2
0020    766 ;
0020    767 ;       This subtest checks for one bit in the odd parity field stuck at zero.
0020    768 ;
0020    769 ;        1. Places ones onto the CPU control store lines.
0020    770 ;        2. Examine the front panel register for a parity error.
0020    771 ;        3. If there is a parity error, skip to subtest 3.
0020    772 ;        4. Prepare a loop of 40 iterations.
0020    773 ;        5. Clear a bit in the odd parity field, indexed by the loop count.
0020    774 ;        6. Is there a parity error?
0020    775 ;        7. If not, print out the loop count to identify the bad bit.
0020    776 ;        8. Otherwise, continue looping.
0020    777 ;
0020    778 ;       Subtest 3
0020    779 ;
0020    780 ;       This subtest checks for one bit in the even parity field stuck at one.
0020    781 ;
0020    782 ;        1. Drive a single one in the odd parity field and all zeroes in the
0020    783 ;           even parity field from DCS into the CPU.
0020    784 ;        2. Examine the front panel register in the RDM for a parity error.
0020    785 ;        3. If there is no parity error, skip to subtest 4.
0020    786 ;        4. Prepare a loop of 40 iterations.
0020    787 ;        5. Leaving a one in the odd parity field, float a one through the
0020    788 ;           even parity field, indexed by the loop count.
0020    789 ;        6. Is there a parity error?
0020    790 ;        7. If yes, the print out the loop count to identify the bad bit.
0020    791 ;        8. Otherwise, continue looping.
0020    792 ;
```

```
                                0020    793 ;       Subtest 4
                                0020    794 ;
                                0020    795 ;       This subtest checks for one bit stuck at zero in the even parity field.
                                0020    796 ;
                                0020    797 ;       1. Drive a single one in the odd parity field and all ones in the even
                                0020    798 ;          parity field from DCS into the CPU.
                                0020    799 ;       2. Examine the front panel register in the RDM for a parity error.
                                0020    800 ;       3. If there is no parity error, skip to subtest 5.
                                0020    801 ;       4. Prepare a loop of 40 iterations.
                                0020    802 ;       5. Leaving a one in the odd parity field, float a one through the even
                                0020    803 ;          parity field, indexed by the loop count.
                                0020    804 ;       6. Is there a parity error?
                                0020    805 ;       7. If yes, the print out the loop count to identify the bad bit.
                                0020    806 ;       8. Otherwise, continue looping.
                                0020    807 ;
                                0020    808 ;       Subtest 5
                                0020    809 ;
                                0020    810 ;       This subtest drives eleven special patterns into the CPU in order to
                                0020    811 ;       to detect bad parity.
                                0020    812 ;
                                0020    813 ;       1. Prepare a loop of 11 iterations.
                                0020    814 ;       2. Load DCS with a test pattern indexed by the loop count.
                                0020    815 ;       3. Do a masked comparison on the contents of the front panel register
                                0020    816 ;          with a table of expected parities.
                                0020    817 ;       4. If the parity bit from the front panel register on the RDM does not
                                0020    818 ;          agree with the expected parity, signal an error.
                                0020    819 ;       5. Otherwise, continue the loop.
                                0020    820 ;       6. Exit.
                                0020    821 ;
                                0020    822 ; DATA PATTERNS
                                0020    823 ;
                                0020    824 ;       Subtest 1       00000000000000000000
                                0020    825 ;
                                0020    826 ;       Subtest 2       FFFFFFFFFFFFFFFFFFFF
                                0020    827 ;
                                0020    828 ;       Subtest 3       80000000000000000000
                                0020    829 ;
                                0020    830 ;       Subtest 4       C07FFFFFFFD80000000
                                0020    831 ;
                                0020    832 ;       Subtest 5       00000000000000000000    ; loop count = 1
                                0020    833 ;                       FFFFFFFFFFFFFFFFFFFF    ; loop count = 2
                                0020    834 ;                       AAAAAAAAAAAAAAAAAAAA    ; loop count = 3
                                0020    835 ;                       55555555555555555555    ; loop count = 4
                                0020    836 ;                       33333333333333333333    ; loop count = 5
                                0020    837 ;                       0F0F0F0F0F0F0F0F0F0F    ; loop count = 6
                                0020    838 ;                       00FF00FF00FF00FF00FF    ; loop count = 7
                                0020    839 ;                       0000FFFF0000FFFF0000    ; loop count = 8
                                0020    840 ;                       00000000FFFFFFFF0000    ; loop count = 9
                                0020    841 ;                       00000000000000FFFF    ; loop count = A
                                0020    842 ;                       88003640200047018O1    ; loop count = B, a CPU no-op
                                0020    843 ;
                                0020    844 ; LOGIC DESCRIPTION
                                0020    845 ;
                                0020    846 ;       The control store lines run through the backplane to the following
                                0020    847 ;       modules in the CPU:
```

```
0020   ;
0020   ;                FPA     DPM     MIC     UBI     CCS     RDM
0020   ;
0020   ; ASSUMPTIONS
0020   ;
0020   ;       This test makes no assumptions about the CPU. However, it does
0020   ;       assume that the RDM is functional.
0020   ;
0020   ; ERROR DESCRIPTION
0020   ;
0020   ;       Below is a table of possible error conditions and their most likely
0020   ;       causes. Whenever there is a failure in Subtests 1-4 there should also
0020   ;       be a failure in Subtest 5.
0020   ;
0020   ;               Error                        Probable Cause
0020   ;       +-----------------------+-----------------------------------------+
0020   ;       | There is one error in | There is a bit stuck at zero or one in|
0020   ;       | any single subtest.   | the parity field under test.           |
0020   ;       +-----------------------+-----------------------------------------+
0020   ;       | There is one error    | There is a double bit error across the|
0020   ;       | in each of Subtests 2 | odd and even parity fields. Use the   |
0020   ;       | and 4.                | loop counts from each subtest to      |
0020   ;       |                       | identify the bad bits.                |
0020   ;       +-----------------------+-----------------------------------------+
0020   ;       | Only Subtest 5 fails. | There is a double bit error in one    |
0020   ;       |                       | parity field.                         |
0020   ;       +-----------------------+-----------------------------------------+
0020   ;
0020   ;--
0020   ;*******************************************************************************
0020
0020           SUBTEST         1
0023
0023   ;///////////////////////////////////////////////////////////////////////////
0023   ;+
0023   ; BIT IN ODD PARITY FIELD STUCK AT ONE
0023   ;
0023   ; This subtest loads the odd and even parity fields with zeros. It then checks
0023   ; for a parity error. If it gets one it continues to subtest 2, otherwise it
0023   ; assumes there is a bit stuck at one in the odd parity field.
0023   ;
0023   ; When this subtest fails it tries to isolate the bad bit in the odd parity
0023   ; field by floating a one through a field of zeros. When the parity error goes
0023   ; away it assumes it has found the bad bit. You can tell the bad bit by
0023   ; indexing into the following table:
0023   ;
0023   ;      Loop  CS Bit #   CS Mnemonic    Loop  CS Bit #   CS Mnemonic
0023   ;
0023   ;       01      00      NEXT 00         15      20      BUS 00
0023   ;       02      01      NEXT 01         16      21      BUS 01
0023   ;       03      02      NEXT 02         17      22      BUS 02
0023   ;       04      03      NEXT 03         18      23      BUS 03
0023   ;       05      04      NEXT 04         19      24      BUS 04
0023   ;       06      05      NEXT 05         1A      25      WCTRL 00
0023   ;       07      06      NEXT 06         1B      26      WCTRL 01
```

```
                                0023    901 ;       08      07      NEXT 07         1C      27      WCTRL 02
                                0023    902 ;       09      08      NEXT 08         1D      28      WCTRL 03
                                0023    903 ;       0A      09      NEXT 09         1E      29      WCTRL 04
                                0023    904 ;       0B      10      NEXT 10         1F      30      WCTRL 05
                                0023    905 ;       0C      11      NEXT 11         20      33      ISTRM
                                0023    906 ;       0D      12      NEXT 12         21      71      MISC 00
                                0023    907 ;       0E      13      NEXT 13         22      72      MISC 01
                                0023    908 ;       0F      14      JSR             23      73      MISC 02
                                0023    909 ;       10      15      CLKX            24      74      MISC 03
                                0023    910 ;       11      16      FPA 00          25      75      MISC 04
                                0023    911 ;       12      17      FPA 01          26      76      LIT 00
                                0023    912 ;       13      18      FPA 02          27      77      LIT 01
                                0023    913 ;       14      19      FPA 03          28      79      PAR 2
                                0023    914 ;-
                                0023    915             LDFIELD         20$..,28$,0,79....,NP   ; SCRAMBLE THE FIRST MICRO WORD
                                0034    916             LOADDCS         28$,,0,1                ; LOAD IT INTO DCS
                                003B    917             LOADREG         DCSAD,15$,,B            ; RESET THE DCS ADDRESS TO 0
                                0043    918             LOADREG         CSAMR,10$,,W            ; LOAD 1800 INTO THE MATCH REG.
                                004B    919             LOADREG         DCSCR,11$,,B            ; SET MICRO_ADD_INH AND PAR_CHK
                                0053    920             LOADREG         RDCTRL,14$,,B           ; PULSE DC_LOW TO LATCH PARITY
                                005B    921             LOADREG         RDCTRL,15$,,B           ; ...
                                0063    922             CMPREGMSK       FRONT1,30$,,30$,,B      ; IS THERE A PARITY ERROR?
                                006F    923             SKIP            2$,NOERROR              ; IF SO BRANCH TO SUBTEST 2
                                0076    924             LOOP            I,1,40                  ; ELSE ODD BIT STUCK AT ONE
                                007D    925             LDFIELD         27$..,28$,24$,24$,,I,I,NP ; LOAD A ONE INTO ZEROS
                                008E    926             LOADDCS         28$,,0,1                ; LOAD TEST PATTERN INTO DCS
                                0095    927             ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                                0097    928             LOADREG         DCSAD,15$,,B            ; RESET THE DCS ADDRESS TO 0
                                009F    929             LOADREG         CSAMR,10$,,W            ; LOAD 1800 INTO THE MATCH REG.
                                00A7    930             LOADREG         DCSCR,11$,,B            ; SET MICRO_ADD_INH AND PAR_CHK
                                00AF    931             LOADREG         RDCTRL,14$,,B           ; PULSE DC_LOW TO LATCH PARITY
                                00B7    932             LOADREG         RDCTRL,15$,,B           ; ...
                                00BF    933             CMPREGMSK       FRONT1,30$,,30$,,B      ; DID THE PARITY ERROR GO AWAY?
                                00CB    934             IFERROR         ,,<<FPA><DPM><MIC><UBI><CCS><RDM>> ; IF SO CALL ERROR
                                00D6    935             LDFIELD         26$..,28$,24$,24$,,I,I,NP ; RESET THE ONE TO A ZERO
                                00E7    936             ENDLOOP         I                       ; LOOP FOR REST OF ODD FIELD
                                00EA    937
                                00EA    938 2$:         SUBTEST         2
                                00ED
                                00ED        ;////////////////////////////////////////////////////////////////////////////////
                                00ED    939 ;+
                                00ED    940 ; BIT IN ODD PARITY FIELD STUCK AT ZERO
                                00ED    941 ;
                                00ED    942 ; This subtest loads the odd and even parity fields with ones. It the checks
                                00ED    943 ; for a parity error. If it gets one it continues to subtest 3, otherwise it
                                00ED    944 ; assumes there is a bit stuck at zero in the odd parity field.
                                00ED    945 ;
                                00ED    946 ; On failure this subtest tries to isolate the bad bit in the odd parity field
                                00ED    947 ; by floating a zero through a field of ones. When the parity error signal goes
                                00ED    948 ; away it will type out an error message with the loop count. You can find the
                                00ED    949 ; bad bit by indexing into the following table:
                                00ED    950 ;
                                00ED    951 ;       Loop  CS Bit #   CS Mnemonic    Loop  CS Bit #   CS Mnemonic
                                00ED    952 ;
                                00ED    953 ;        01      00      NEXT 00         15      20      BUS 00
```

```
                                00ED   954 ;       02      01      NEXT 01         16      21      BUS 01
                                00ED   955 ;       03      02      NEXT 02         17      22      BUS 02
                                00ED   956 ;       04      03      NEXT 03         18      23      BUS 03
                                00ED   957 ;       05      04      NEXT 04         19      24      BUS 04
                                00ED   958 ;       06      05      NEXT 05         1A      25      WCTRL 00
                                00ED   959 ;       07      06      NEXT 06         1B      26      WCTRL 01
                                00ED   960 ;       08      07      NEXT 07         1C      27      WCTRL 02
                                00ED   961 ;       09      08      NEXT 08         1D      28      WCTRL 03
                                00ED   962 ;       0A      09      NEXT 09         1E      29      WCTRL 04
                                00ED   963 ;       0B      10      NEXT 10         1F      30      WCTRL 05
                                00ED   964 ;       0C      11      NEXT 11         20      33      ISTRM
                                00ED   965 ;       0D      12      NEXT 12         21      71      MISC 00
                                00ED   966 ;       0E      13      NEXT 13         22      72      MISC 01
                                00ED   967 ;       0F      14      JSR             23      73      MISC 02
                                00ED   968 ;       10      15      CLKX            24      74      MISC 03
                                00ED   969 ;       11      16      FPA 00          25      75      MISC 04
                                00ED   970 ;       12      17      FPA 01          26      76      LIT 00
                                00ED   971 ;       13      18      FPA 02          27      77      LIT 01
                                00ED   972 ;       14      19      FPA 03          28      79      PAR 2
                                00ED   973 ;-
                                00ED   974              LDFIELD         21$,,,28$,0,79,,,,NP    ; SCRAMBLE THE TEST MICRO-WORD
                                00FE   975              LOADDCS         28$,,0,1                ; LOAD IT INTO DCS
                                0105   976              LOADREG         DCSAD,15$,,B            ; RESET THE DCS ADRESS TO 0
                                010D   977              LOADREG         CSAMR,10$,,W            ; LOAD 1800 INTO THE MATCH REG.
                                0115   978              LOADREG         DCSCR,11$,,B            ; SET MICRO_ADD_INH AND PAR_CHK
                                011D   979              LOADREG         RDCTRL,14$,,B           ; PULSE DC_LOW TO LATCH PARITY
                                0125   980              LOADREG         RDCTRL,15$,,B           ; ...
                                012D   981              CMPREGMSK       FRONT1,30$,,30$,,B      ; IS THERE A PARITY ERROR?
                                0139   982              SKIP            3$,NOERROR              ; IF SO GOTO SUBTEST 3
                                0140   983              LOOP            I,1,40                  ; ELSE ODD BIT STUCK AT ZERO
                                0147   984              LDFIELD         26$,,,28$,24$,24$,,I,I,NP ; LOAD A ZERO INTO ONES
                                0158   985              LOADDCS         28$,,0,1                ; LOAD TEST MICRO-WORD
                                015F   986              ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                                0161   987              LOADREG         DCSAD,15$,,B            ; RESET THE DCS ADDRESS TO ZERO
                                0169   988              LOADREG         CSAMR,10$,,W            ; LOAD 1800 INTO THE MATCH REG.
                                0171   989              LOADREG         DCSCR,11$,,B            ; SET MICRO_ADD_INH AND PAR_CHK
                                0179   990              LOADREG         RDCTRL,14$,,B           ; PULSE DC_LOW TO LATCH PARITY
                                0181   991              LOADREG         RDCTRL,15$,,B           ; ...
                                0189   992              CMPREGMSK       FRONT1,30$,,30$,,B      ; IS THERE A PARITY ERROR?
                                0195   993              IFERROR         ,,<<FPA><DPM><MIC><UBI><CCS><RDM>> ; IF SO CALL ERROR
                                01A0   994              LDFIELD         27$,,,28$,24$,24$,,I,I,NP ; RESET THE ZERO TO A ONE
                                01B1   995              ENDLOOP         I                       ; LOOP FOR REST OF ODD FIELD
                                01B4   996
                                01B4   997 3$:          SUBTEST         3
                                01B7
                                01B7       ;////////////////////////////////////////////////////////////////////////////////
                                01B7   998 ;+
                                01B7   999 ; BIT IN EVEN PARITY FIELD STUCK AT ONE
                                01B7  1000 ;
                                01B7  1001 ; This subtest loads the odd parity field with a single one and the even parity
                                01B7  1002 ; field with zeros. It then checks for no parity error. If there is none it
                                01B7  1003 ; skips to subtest 4, otherwise it assumes there is a bit stuck at one in the
                                01B7  1004 ; even parity field.
                                01B7  1005 ;
                                01B7  1006 ; When this subtest fails it tries to isolate the bad bit in the odd parity
```

```
                        ; field by floating a one through a field of zeros. When the parity error
                        ; signal goes away it assumes it has found the bad bit. You can tell the bad
                        ; bit by indexing into the following table:
                        ;
                        ;      Loop  CS Bit #   CS Mnemonic    Loop  CS Bit #   CS Mnemonic
                        ;
                        ;       01      31       CC 00          15      52       ALPCTL 04
                        ;       02      32       CC 01          16      53       ALPCTL 05
                        ;       03      34       RSRC 00        17      54       ALPCTL 06
                        ;       04      35       RSRC 01        18      55       ALPCTL 07
                        ;       05      36       RSRC 02        19      56       ALPCTL 08
                        ;       06      37       RSRC 03        1A      57       ALPCTL 09
                        ;       07      38       RSRC 04        1B      58       ROT 00
                        ;       08      39       RSRC 05        1C      59       ROT 01
                        ;       09      40       DTYPE 00       1D      60       ROT 02
                        ;       0A      41       DTYPE 01       1E      61       ROT 03
                        ;       0B      42       BUT 00         1F      62       ROT 04
                        ;       0C      43       BUT 01         20      63       ROT 05
                        ;       0D      44       BUT 02         21      64       MSRC 00
                        ;       0E      45       BUT 03         22      65       MSRC 01
                        ;       0F      46       BUT 04         23      66       MSRC 02
                        ;       10      47       BUT 05         24      67       MSRC 03
                        ;       11      48       ALPCTL 00      25      68       MSRC 04
                        ;       12      49       ALPCTL 01      26      69       SPW 00
                        ;       13      50       ALPCTL 02      27      70       SPW 01
                        ;       14      51       ALPCTL 03      28      78       PAR 1
                        ;-
                        LDFIELD         22$...,28$.0,79,...,NP    ; SCRAMBLE THE TEST MICRO-WORD
                        LOADDCS         28$.,0,1                  ; LOAD IT INTO DCS
                        LOADREG         DCSAD,15$..B              ; RESET THE DCS ADDRESS TO 0
                        LOADREG         CSAMR,10$..W              ; LOAD 1800 INTO THE MATCH REG.
                        LOADREG         DCSCR,11$,.B              ; SET MICRO_ADD_INH AND PAR_CHK
                        LOADREG         RDCTRL,14$..B             ; PULSE DC_LOW TO LATCH PARITY
                        LOADREG         RDCTRL,15$..B             ; ...
                        CMPREGMSK       FRONT1,31$.,30$..B        ; IS THERE A PARITY ERROR?
                        SKIP            4$,NOERROR                ; IF NOT THEN GOTO SUBTEST 4
                        LOOP            I,1,40                    ; ELSE EVEN BIT STUCK AT ONE
                        LDFIELD         27$...,28$.25$.25$..I.I.NP ; LOAD A ONE INTO ZEROS
                        LOADDCS         28$.,0,1                  ; LOAD TEST PATTERN INTO DCS
                        ERRLOOP                                   ; LOOP ON ERROR FROM HERE
                        LOADREG         DCSAD,15$..B              ; RESET THE DCS ADDRESS TO 0
                        LOADREG         CSAMR,10$..W              ; LOAD 1800 INTO THE MATCH REG.
                        LOADREG         DCSCR,11$,.B              ; SET MICRO_ADD_INH AND PAR_CHK
                        LOADREG         RDCTRL,14$..B             ; PULSE DC_LOW TO LATCH PARITY
                        LOADREG         RDCTRL,15$..B             ; ...
                        CMPREGMSK       FRONT1,31$.,30$.,B        ; IS THERE A PARITY ERROR?
                        IFERROR         .,<<FPA><DPM><MIC><UBI><CCS><RDM>> ; IF SO CALL ERROR
                        LDFIELD         26$...,28$.25$,25$..I.I.NP ; RESET THE ONE TO A ZERO
                        ENDLOOP         I                         ; LOOP FOR REST OF EVEN FIELD
                        ;
4$:                     SUBTEST         4
                        ;
                        ;;;
```

```
                    1060 ;
                    1061 ; This subtest loads the odd parity field with a single one and the even parity
                    1062 ; field with all ones. It then checks for no parity error. If there is none it
                    1063 ; skips to the next test. Otherwise it assumes there is a bit stuck at zero in
                    1064 ; the even parity field.
                    1065 ;
                    1066 ; When this subtest fails it tries to isolate the bad bit in the odd parity
                    1067 ; field by floating a zero through a field of ones. When the parity error
                    1068 ; signal goes away it assumes it has found the bad bit. You can tell the bad
                    1069 ; bit by indexing into the following table:
                    1070 ;
                    1071 ;      Loop  CS Bit #   CS Mnemonic    Loop  CS Bit #   CS Mnemonic
                    1072 ;
                    1073 ;        01      31      CC 00          15      52      ALPCTL 04
                    1074 ;        02      32      CC 01          16      53      ALPCTL 05
                    1075 ;        03      34      RSRC 00        17      54      ALPCTL 06
                    1076 ;        04      35      RSRC 01        18      55      ALPCTL 07
                    1077 ;        05      36      RSRC 02        19      56      ALPCTL 08
                    1078 ;        06      37      RSRC 03        1A      57      ALPCTL 09
                    1079 ;        07      38      RSRC 04        1B      58      ROT 00
                    1080 ;        08      39      RSRC 05        1C      59      ROT 01
                    1081 ;        09      40      DTYPE 00       1D      60      ROT 02
                    1082 ;        0A      41      DTYPE 01       1E      61      ROT 03
                    1083 ;        0B      42      BUT 00         1F      62      ROT 04
                    1084 ;        0C      43      BUT 01         20      63      ROT 05
                    1085 ;        0D      44      BUT 02         21      64      MSRC 00
                    1086 ;        0E      45      BUT 03         22      65      MSRC 01
                    1087 ;        0F      46      BUT 04         23      66      MSRC 02
                    1088 ;        10      47      BUT 05         24      67      MSRC 03
                    1089 ;        11      48      ALPCTL 00      25      68      MSRC 04
                    1090 ;        12      49      ALPCTL 01      26      69      SPW 00
                    1091 ;        13      50      ALPCTL 02      27      70      SPW 01
                    1092 ;        14      51      ALPCTL 03      28      78      PAR 1
                    1093 ;-
0281                1094            LDFIELD         23$...,28$.0,79....,NP      ; SCRAMBLE THE TEST MICRO-WORD
0292                1095            LOADDCS         28$.,0,1                    ; LOAD IT INTO DCS
0299                1096            LOADREG         DCSAD,15$..B                ; RESET THE DCS ADDRESS TO ZERO
02A1                1097            LOADREG         CSAMR,10$..W                ; LOAD 1800 INTO THE MATCH REG.
02A9                1098            LOADREG         DCSCR,11$..B                ; SET MICRO_ADD_INH AND PAR_CHK
02B1                1099            LOADREG         RDCTRL,14$..B               ; PULSE DC_LOW TO LATCH PARITY
02B9                1100            LOADREG         RDCTRL,15$..B               ; ...
02C1                1101            CMPREGMSK       FRONT1,31$..30$..B          ; IS THERE A PARITY ERROR?
02CD                1102            SKIP            5$,NOERROR                  ; IF NOT GOTO SUBTEST 5
02D4                1103            LOOP            I,1,40                      ; ELSE EVEN BIT STUCK AT ZERO
02DB                1104            LDFIELD         26$...,28$.25$.25$..I.I.NP  ; LOAD A ZERO INTO ONES
02EC                1105            LOADDCS         28$..0.1                    ; LOAD TEST PATTERN INTO DCS
02F3                1106            ERRLOOP                                     ; LOOP ON ERROR FROM HERE
02F5                1107            LOADREG         DCSAD.15$..B                ; RESET THE DCS ADDRESS TO ZERO
02FD                1108            LOADREG         CSAMR.10$..W                ; LOAD 1800 INTO THE MATCH REG.
0305                1109            LOADREG         DCSCR.11$..B                ; SET MICRO_ADD_INH AND PAR_CHK
030D                1110            LOADREG         RDCTRL.14$..B               ; PULSE DC_LOW TO LATCH PARITY
0315                1111            LOADREG         RDCTRL.15$..B               ; ...
031D                1112            CMPREGMSK       FRONT1.31$..30$..B          ; IS THERE A PARITY ERROR?
0329                1113            IFERROR         ,,<<FPA><DPM><MIC><UBI><CCS><RDM>> ; IF SO CALL ERROR
0334                1114            LDFIELD         27$...28$.25$.25$..I.I.NP ; RESET THE ZERO TO A ONE
```

```
                        0345  1115          ENDLOOP         I                       ; LOOP FOR REST OF EVEN FIELD
                        0348  1116
                        0348  1117 5$:      SUBTEST         5
                        034B
                        034B          ;//////////////////////////////////////////////////////////////////////////////
                        034B  1118 ;+
                        034B  1119 ; DOUBLE BIT ERROR DETECTION
                        034B  1120 ;
                        034B  1121 ; This subtest will detect two bits stuck high or low or tied to each other
                        034B  1122 ; by looping through eleven special patterns.
                        034B  1123 ;
                        034B  1124 ;-
                        034B  1125          LOOP            I,1,11                  ; PREPARE A LOOP OF 11
                        0352  1126          LDFIELD         40$,I,M,28$,0,79,,,,NP  ; SCRAMBLE THE TEST PATTERN
                        0363  1127          LOADDCS         28$,,0,1                ; LOAD SCRAMBLED PATTERN
                        036A  1128          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        036C  1129          LOADREG         DCSAD,15$,,B            ; RESET THE DCS ADDRESS TO ZERO
                        0374  1130          LOADREG         CSAMR,10$,,W            ; DCS ADDRESS 1800 TO MATCH REG
                        037C  1131          LOADREG         DCSCR,11$,,B            ; DCS AND MATCH REG TO CS BUS
                        0384  1132          LOADREG         RDCTRL,14$,,B           ; PULSE DC_LOW TO LATCH PARITY
                        038C  1133          LOADREG         RDCTRL,15$,,B           ; ...
                        0394  1134          CMPREGMSK       FRONT1,30$,I,30$,,B     ; IS THE PARITY CORRECT?
                        03A0  1135          IFERROR         ,,<<FPA><DPM><MIC><UBI><CCS><RDM>> ; IF SO CALL ERROR
                        03AB  1136          ENDLOOP         I                       ; CONTINUE THE LOOP
                        03AE  1137          LOADREG         DCSCR,13$,,B            ; DISABLE DCS AND MATCH REG.
                        03B6  1138          SKIP                                    ; END OF TEST
                        03BD  1139
                        03BD  1140          BEGIN_TEST_DATA
                        03BD
                        03BD          ;-------------------------------------------------------------------------
                        03BD  1141 ;
                        03BD  1142 ; RDM AND DCS CONTROL DATA
                        03BD  1143 ;
                        03BD  1144
                1800    03BD  1145 10$:     .WORD   DCS_START
                  E8    03BF  1146 11$:     .BYTE   MICRO_ADDR_INH!PAR_CHK_ENABL!CLK_CTRL_0!CLK_CTRL_1
                  60    03C0  1147 13$:     .BYTE   CLK_CTRL_0!CLK_CTRL_1
                  40    03C1  1148 14$:     .BYTE   DC_LOW
                  00    03C2  1149 15$:     .BYTE   0
                        03C3  1150
                        03C3  1151 ;
                        03C3  1152 ; DATA FOR SUBTESTS 1-4
                        03C3  1153 ;
                        03C3  1154
00 00 00 00 00 00 00 00 03C3  1155 20$:     .BYTE   ^X00,^X00,^X00,^X00,^X00,^X00,^X00,^X00,^X00,^X00 ; BAD PARITY
FF FF FF FF FF FF FF FF 03CD  1156 21$:     .BYTE   ^XFF,^XFF,^XFF,^XFF,^XFF,^XFF,^XFF,^XFF,^XFF,^XFF ; BAD PARITY
80 00 00 00 00 00 00 00 03D7  1157 22$:     .BYTE   ^X00,^X00,^X00,^X00,^X00,^X00,^X00,^X00,^X00,^X80 ; GOOD PARITY
C0 7F FF FF FF FD 80 00 03E1  1158 23$:     .BYTE   ^X00,^X00,^X00,^X80,^XFD,^XFF,^XFF,^XFF,^X7F,^XC0 ; GOOD PARITY
                        03EB  1159
                        03EB  1160 ;
                        03EB  1161 ; BIT MAPS FOR FLOATING PATTERNS
                        03EB  1162 ;
                        03EB  1163
07 06 05 04 03 02 01 00 03EB  1164 24$:     .BYTE   00,01,02,03,04,05,06,07         ; ODD PARITY FIELD
0F 0E 0D 0C 0B 0A 09 08 03F3  1165          .BYTE   08,09,10,11,12,13,14,15
```

```
                17 16 15 14 13 12 11 10  03FB  1166          .BYTE   16,17,18,19,20,21,22,23
                21 1E 1D 1C 1B 1A 19 18  0403  1167          .BYTE   24,25,26,27,28,29,30,33
                4F 4D 4C 4B 4A 49 48 47  040B  1168          .BYTE   71,72,73,74,75,76,77,79
                                         0413  1169
                27 26 25 24 23 22 20 1F  0413  1170 25$:     .BYTE   31,32,34,35,36,37,38,39          ; EVEN PARITY FIELD
                2F 2E 2D 2C 2B 2A 29 28  041B  1171          .BYTE   40,41,42,43,44,45,46,47
                37 36 35 34 33 32 31 30  0423  1172          .BYTE   48,49,50,51,52,53,54,55
                3F 3E 3D 3C 3B 3A 39 38  042B  1173          .BYTE   56,57,58,59,60,61,62,63
                4E 46 45 44 43 42 41 40  0433  1174          .BYTE   64,65,66,67,68,69,70,78
                                         043B  1175
                                         043B  1176 ;
                                         043B  1177 ; VALUES TO BE FLOATED
                                         043B  1178 ;
                                         043B  1179
                                      00 043B  1180 26$:     .BYTE   0
                                      01 043C  1181 27$:     .BYTE   1
                                         043D  1182
                                         043D  1183 ;
                                         043D  1184 ; SCRAMBLED MICRO-WORD WITH DCS CONTROL FILE
                                         043D  1185 ;
                                         043D  1186
   FD 00 00 00 00 00 00 00 00 00 00      043D  1187 28$:     .BYTE   ^X00,^X00,^X00,^X00,^X00,^X00,^X00,^X00,^X00,^XFD
                                         0448  1188
                                         0448  1189 ;
                                         0448  1190 ; TABLE OF EXPECTED PARITY
                                         0448  1191 ;
                                         0448  1192
                                      20 0448  1193 30$:     .BYTE   PARITY_ERROR
                                      20 0449  1194          .BYTE   PARITY_ERROR
                                      20 044A  1195          .BYTE   PARITY_ERROR
                                      20 044B  1196          .BYTE   PARITY_ERROR
                                      20 044C  1197          .BYTE   PARITY_ERROR
                                      20 044D  1198          .BYTE   PARITY_ERROR
                                      20 044E  1199          .BYTE   PARITY_ERROR
                                      20 044F  1200          .BYTE   PARITY_ERROR
                                      20 0450  1201          .BYTE   PARITY_ERROR
                                      20 0451  1202          .BYTE   PARITY_ERROR
                                      00 0452  1203 31$:     .BYTE   0
                                         0453  1204
                                         0453  1205 ;
                                         0453  1206 ; DOUBLE BIT ERROR PATTERNS
                                         0453  1207 ;
                                         0453  1208
00 00 00 00 00 00 00 00 00 00            0453  1209 40$:     .BYTE   ^X00,^X00,^X00,^X00,^X00,^X00,^X00,^X00,^X00,^X00
FF FF FF FF FF FF FF FF FF FF            045D  1210          .BYTE   ^XFF,^XFF,^XFF,^XFF,^XFF,^XFF,^XFF,^XFF,^XFF,^XFF
AA AA AA AA AA AA AA AA AA AA            0467  1211          .BYTE   ^XAA,^XAA,^XAA,^XAA,^XAA,^XAA,^XAA,^XAA,^XAA,^XAA
55 55 55 55 55 55 55 55 55 55            0471  1212          .BYTE   ^X55,^X55,^X55,^X55,^X55,^X55,^X55,^X55,^X55,^X55
33 33 33 33 33 33 33 33 33 33            047B  1213          .BYTE   ^X33,^X33,^X33,^X33,^X33,^X33,^X33,^X33,^X33,^X33
0F 0F 0F 0F 0F 0F 0F 0F 0F 0F            0485  1214          .BYTE   ^X0F,^X0F,^X0F,^X0F,^X0F,^X0F,^X0F,^X0F,^X0F,^X0F
FF 00 FF 00 FF 00 FF 00 FF 00            048F  1215          .BYTE   ^X00,^XFF,^X00,^XFF,^X00,^XFF,^X00,^XFF,^X00,^XFF
00 00 FF FF 00 00 FF FF 00 00            0499  1216          .BYTE   ^X00,^X00,^XFF,^XFF,^X00,^X00,^XFF,^XFF,^X00,^X00
00 00 FF FF FF FF 00 00 00 00            04A3  1217          .BYTE   ^X00,^X00,^X00,^X00,^XFF,^XFF,^XFF,^XFF,^X00,^X00
FF FF 00 00 00 00 00 00 00 00            04AD  1218          .BYTE   ^X00,^X00,^X00,^X00,^X00,^X00,^X00,^X00,^XFF,^XFF
88 00 03 64 02 00 04 70 18 01            04B7  1219          .BYTE   ^X01,^X18,^X70,^X04,^X00,^X02,^X64,^X03,^X00,^X88 ; A NO-OP
                                         04C1  1220
```

```
                        04C1  1221 END_TEST_DATA
                        04C1
                        04C1        ;-----------------------------------------------------------------------
                        04C1  1222
                        04C1  1223 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
                        .SBTTL  "TEST   007     Long Literal Register to WBUS Data Path
                1225 ;*****************************************************************************
                1226 ;++
                1227 ;
                1228 ; FUNCTIONAL DESCRIPTION
                1229 ;
                1230 ;       This test checks that the Long Literal Register and the RBUS
                1231 ;       path thru the ALP Gate Arrays to the WBUS are operational.
                1232 ;
                1233 ; TEST ALGORITHM
                1234 ;
                1235 ;       1. Load the micro words to do the test.
                1236 ;       2. Execute the micro sequence.
                1237 ;               1. Write LONLIT register with 55555555.
                1238 ;               2. Read LONLIT register into RDM.
                1239 ;       3. Check the results of the Long Literal register read.
                1240 ;
                1241 ; DATA PATTERN
                1242 ;
                1243 ;       The following pattern is used to test the LONLIT Register:
                1244 ;
                1245 ;               55555555
                1246 ;
                1247 ; LOGIC DESCRIPTION
                1248 ;
                1249 ;       This test checks the latches that hold the LONG LITERAL,the
                1250 ;       Literal register write signal(CLA GATE ARRAY),the Literal
                1251 ;       register read signal(SPA GATE ARRAY),the RBUS path (ALK GATE
                1252 ;       ARRAY) thru the ALP Gate Arrays to the WBUS and the D CLK L
                1253 ;       signal from the SAC Gate Array.
                1254 ;       THIS IS THE FIRST TEST THAT ACTUALLY TRIES TO EXECUTE MICROCODE.
                1255 ;       IF THE TEST ENDS WITH A MICROTRAP ADDRESS THEN TRY REPLACING THE
                1256 ;       MIC AND FPA MODULES ALONG WITH DPM UNTIL YOU DETERMINE WHO'S CAUSING
                1257 ;       THE TRAP CONDITION.
                1258 ;
                1259 ; ASSUMPTIONS
                1260 ;
                1261 ;       This test requires no signals from any other CPU modules but
                1262 ;       assumes an operational RBUS and WBUS.
                1263 ;
                1264 ; ERROR DESCRIPTION
                1265 ;
                1266 ;       Data expected:55555555
                1267 ;       Data received:
                1268 ;
                1269 ;--
                1270 ;*****************************************************************************
                1271 ;/////////////////////////////////////////////////////////////////////////////
                1272
002C            1273           INITIALIZE                                   ;DCLO low,Init scratchpads
002E            1274           EXPUTRAP                                     ;POSSIBLE MICROTRAP
0030            1275           LOADDCS 1$,,0,4                              ;Load the test micro instrs.
0037            1276           ERRLOOP                                      ;Set the error loop
0039            1277           FETCH 0                                      ;Get the first micro word
003E            1278           BRSTCLK 3                                    ;Clock should stop location 3
```

```
                     0042  1279          CMPREGMSK       CSTRP,3$,,4$,,W,          ;TEST CS ADDRESS
                     004E  1280          IFERROR         ,<SAC>,<<MIC><FPA>>       ;TEST MICROTRAPPED
                     0056  1281          CMPREG          WHREG,2$,,Long            ;Check the received data
                     005F  1282          IFERROR         ,<<CLA>,<SPA>,<ALP>,<SAC>> ;Continue if no error else
                     0068  1283                                                    ;call out CLA,SPA,ALP,SAC
                     0068  1284          SKIP                                      ;End of test
                     006F  1285
                     006F  1286 BEGIN_TEST_DATA
                     006F
                     006F          ;-----------------------------------------------------------------------
                     006F  1287 ;+
                     006F  1288 ; The following micro words are used to write and read the Long Literal
                     006F  1289 ; register with the 55555555 data pattern.
                     006F  1290 ;-
                     006F  1291
                     006F  1292 1$:
U 00, 7700,C800,0364,0300,0470,1801  006F  1293 ;% 00    NOP                              ;NOP micro instruction
U 01, 7700,B800,5555,5555,0470,1802  007A  1297 ;% 01    LONLIT_[55555555]                ;LIT/LONLIT
U 02, 6700,0800,0047,03D4,0470,1803  0085  1301 ;% 02    RDM_LONLIT                       ;Load results into RDM
U 03, 7300,C800,0364,0300,0470,1804  0090  1305 ;% 03    NOP,STOP.CLOCK                   ;Stop clock and check results
                          55555555   009B  1309 2$:      .Long ^x55555555                 ;Expected received data
                              1803   009F  1310 3$:      .WORD   ^X1803                   ;LAST DCS ADDRESS
                              3FFF   00A1  1311 4$:      .WORD   ^X3FFF                   ;MASK FOR CMPREGMSK PSEUDO OP
                     00A3  1312
                     00A3  1313 END_TEST_DATA
                     00A3
                     00A3          ;-----------------------------------------------------------------------
                     00A3  1314
                     00A3  1315 ENDTEST
```

```
                0029        .SBTTL  "TEST   008     Long Literal Register Data Integrity
                0029  1317 ;*******************************************************************************
                0029  1318 ;++
                0029  1319 ;
                0029  1320 ; FUNCTIONAL DESCRIPTION
                0029  1321 ;
                0029  1322 ;       This test checks the data integrity of the LONG LITERAL
                0029  1323 ;       REGISTER by loading and reading 6 patterns thru the register.
                0029  1324 ;
                0029  1325 ; TEST ALGORITHM
                0029  1326 ;
                0029  1327 ;       1. Load the micro words to do the test.
                0029  1328 ;       2. Setup a loop of six iterations.
                0029  1329 ;       3. Select and load a data pattern into LONLIT Field of micro
                0029  1330 ;               word.
                0029  1331 ;       4. Load DCS and execute the micro sequence.
                0029  1332 ;               1. Load LONLIT with pattern.
                0029  1333 ;               2. Read LONLIT into RDM.
                0029  1334 ;       5. Check the results.
                0029  1335 ;       6. Looped six times?If not go to step three.
                0029  1336 ;
                0029  1337 ;
                0029  1338 ; DATA PATTERNS
                0029  1339 ;
                0029  1340 ;       One of the following patterns is used each time thru the loop:
                0029  1341 ;
                0029  1342 ;               AAAAAAAA
                0029  1343 ;               55555555
                0029  1344 ;               33333333
                0029  1345 ;               0F0F0F0F
                0029  1346 ;               00FF00FF
                0029  1347 ;               0000FFFF
                0029  1348 ;
                0029  1349 ; LOGIC DESCRIPTION
                0029  1350 ;
                0029  1351 ;       This test checks the data integrity of the latches that hold
                0029  1352 ;       the LONG LITERAL and the integrity of the data path thru the
                0029  1353 ;       ALP GATE ARRAYS on the DPM module.
                0029  1354 ;
                0029  1355 ; ASSUMPTIONS
                0029  1356 ;
                0029  1357 ;       This test requires no signals from any other CPU modules but
                0029  1358 ;       assumes an operational RBUS AND WBUS.
                0029  1359 ;
                0029  1360 ; ERROR DESCRIPTION
                0029  1361 ;
                0029  1362 ;       Data expected: Same as pattern loaded
                0029  1363 ;       Data received:
                0029  1364 ;       Loop Count:
                0029  1365 ;--
                0029  1366 ;*******************************************************************************
                0029  1367 ;///////////////////////////////////////////////////////////////////////////////
                0029  1368
                0029  1369          INITIALIZE                          ;DCLO Low,Init scratchpads
                002B  1370          LOOP    I,1,6                       ;Initialize the loop count
```

```
                        0032  1371          SA01PAT I,2$,Longlit,3$          ;Select a test pattern
                        003A  1372                                           ;for the LONLIT field
                        003A  1373          LOADDCS 1$,,0,4                  ;Load the test micro instrs
                        0041  1374          ERRLOOP                          ;Set the error loop
                        0043  1375          FETCH 0                          ;Get the first micro word
                        0048  1376          BRSTCLK 3                        ;Clock should stop location 3
                        004C  1377          CMPREG  WHREG,3$,,Long           ;Check the received data
                        0055  1378          IFERROR ,<<CLA>,<SPA>,<ALP>>     ;Continue if no error
                        005D  1379                                           ;else call out CLA,SPA,ALP
                        005D  1380          ENDLOOP I                        ;Continue with the loop
                        0060  1381          SKIP
                        0067  1382
                        0067  1383 BEGIN_TEST_DATA
                        0067
                        0067        ;-------------------------------------------------------------------------
                        0067  1384 ;+
                        0067  1385 ; The following micro words are used to write and read the LONG
                        0067  1386 ;  LITERAL register with six data patterns.
                        0067  1387 ;-
                        0067  1388
                        0067  1389 1$:
U 00, 7700,C800,0364,0300,0470,1801     0067  1390 ;% 00    NOP                          ;NOP Micro instruction
                        0072  1394 2$:
U 01, 7700,7800,7FFF,FFFF,8470,1802     0072  1395 ;% 01    LONLIT_[0]                   ;Test pattern goes here
U 02, 6700,0800,0047,03D4,0470,1803     007D  1399 ;% 02    RDM_LONLIT                   ;Test pattern to RDM
U 03, 7300,C800,0364,0300,0470,1804     0088  1403 ;% 03    NOP,STOP.CLOCK               ;Stop clock and check results
                        0093  1407
                   00000000 0093  1408 3$:      .Long                            ;Expected data goes here.
                        0097  1409
                        0097  1410 END_TEST_DATA
                        0097
                        0097        ;-------------------------------------------------------------------------
                        0097  1411
                        0097  1412 ENDTEST
```

```
                        003D        .SBTTL  "TEST   009     ALK Gate Array,ALP Gate Array Opcode Decode,RBUS-to-WBUS
                        003D  1414 ;*******************************************************************************
                        003D  1415 ;++
                        003D  1416 ;
                        003D  1417 ; FUNCTIONAL DESCRIPTION
                        003D  1418 ;
                        003D  1419 ;       This test checks the ALK Gate Array and ALP Gate ARRAYS for
                        003D  1420 ;       proper opcode decode of RBUS-TO-WBUS data path.
                        003D  1421 ;
                        003D  1422 ; TEST ALGORITHM
                        003D  1423 ;
                        003D  1424 ;       1. Load the micro words to do the test.
                        003D  1425 ;       2. Setup a loop of six iterations.
                        003D  1426 ;       3. Load selected opcode into ALPCTL field of micro word.
                        003D  1427 ;       4. Execute the micro sequence.
                        003D  1428 ;               1. Load LONLIT register with AAAAAAAA.
                        003D  1429 ;               2. Read LONLIT register into RDM.
                        003D  1430 ;       5. Check the resrults of the LONLIT register read.
                        003D  1431 ;       6. Looped six times?If not go to step three.
                        003D  1432 ;
                        003D  1433 ; DATA PATTERNS
                        003D  1434 ;
                        003D  1435 ;       The following pattern is used for this test:
                        003D  1436 ;
                        003D  1437 ;               AAAAAAAA
                        003D  1438 ;
                        003D  1439 ; ALP OPCODES
                        003D  1440 ;
                        003D  1441 ;       The following ALP Control Field OPCODES Are used in this test:
                        003D  1442 ;
                        003D  1443 ;             ALPCTL/WX_R.Q_M
                        003D  1444 ;             ALPCTL/WX_R.Q_XM
                        003D  1445 ;             ALPCTL/WX_R.Q_D
                        003D  1446 ;             ALPCTL/WX_D_R.Q_M
                        003D  1447 ;             ALPCTL/WX_D_R.Q_XM
                        003D  1448 ;             ALPCTL/WX_D_R.Q_D
                        003D  1449 ;
                        003D  1450 ; LOGIC DESCRIPTION
                        003D  1451 ;
                        003D  1452 ;       This test checks that the ALK Gate Array and ALP Gate Arrays
                        003D  1453 ;       can decode the ALPCTL field for the 6 RBUS-TO-WBUS opcodes.
                        003D  1454 ;
                        003D  1455 ; ASSUMPTIONS
                        003D  1456 ;
                        003D  1457 ;       This test requires no signals from any other CPU modules but
                        003D  1458 ;       assumes an operational RBUS and WBUS.
                        003D  1459 ;
                        003D  1460 ; ERROR DESCRIPTION
                        003D  1461 ;
                        003D  1462 ;       Data expected:AAAAAAAA
                        003D  1463 ;       Data received:
                        003D  1464 ;       loop Count:
                        003D  1465 ;
                        003D  1466 ;--
                        003D  1467 ;*******************************************************************************
```

```
                                003D  1468 ;//////////////////////////////////////////////////////////////////////////////
                                003D  1469
                                003D  1470         INITIALIZE                          ;DCLO low,Init. scratchpads
                                003F  1471         LOOP    I,1,6                       ;Init loop count
                                0046  1472         LDFIELD 3$,I,Word,2$,48,57          ;Select and load ALPCTL field
                                0057  1473         LOADDCS 1$,,0,4                     ;Load the test micro instrs.
                                005E  1474         ERRLOOP                             ;Set the error loop
                                0060  1475         FETCH 0                             ;Load the first micro word
                                0065  1476         BRSTCLK 3                           ;Clock stop location 3
                                0069  1477         CMPREG  WHREG,4$,,Long              ;Check the received data
                                0072  1478         IFERROR ,<<ALK>,<ALP>>              ;Continue if no error
                                0079  1479                                             ;else call out ALK,ALP
                                0079  1480         ENDLOOP I                           ;Continue with the loop
                                007C  1481         SKIP                                ;End test
                                0083  1482
                                0083  1483 BEGIN_TEST_DATA
                                0083
                                0083        ;-------------------------------------------------------------------
                                0083  1484 ;+
                                0083  1485 ; The following micro words are used to write and read the LONLIT
                                0083  1486 ;  register and steer the data thru the ALP GATE ARRAYS:
                                0083  1487 ;-
                                0083  1488
                                0083  1489 1$:
U 00, 7700,C800,0364,0300,0470,1801  0083  1490 ;% 00    NOP                            ;NOP micrp instruction
U 01, 7700,7800,2AAA,AAAA,8470,1802  008E  1494 ;% 01    LONLIT_[0AAAAAAAA]             ;Write the LONLIT register
                                0099  1498 2$:
U 02, 6700,0800,0047,03D4,0470,1803  0099  1499 ;% 02    RDM_LONLIT                     ;ALPCTL/SELECTED
U 03, 7300,C800,0364,0300,0470,1804  00A4  1503 ;% 03    NOP,STOP.CLOCK                 ;Stop clock and check results
                                00AF  1507
                            0047  00AF  1508 3$:      .Word ^x47                     ;ALPCTL/WX_R.Q_M
                            0147  00B1  1509          .Word ^x147                    ;ALPCTL/WX_R.Q_XM
                            0247  00B3  1510          .Word ^x247                    ;ALPCTL/WX_R.Q_D
                            0057  00B5  1511          .Word ^x57                     ;ALPCTL/WX_D_R.Q_M
                            0157  00B7  1512          .Word ^x157                    ;ALPCTL/WX_D_R.Q_XM
                            0257  00B9  1513          .Word ^x257                    ;ALPCTL/WX_D_R.Q_D
                                00BB  1514
                       AAAAAAAA  00BB  1515 4$:      .Long ^xAAAAAAAA               ;Expected data goes here
                                00BF  1516
                                00BF  1517 END_TEST_DATA
                                00BF
                                00BF        ;-------------------------------------------------------------------
                                00BF  1518
                                00BF  1519 ENDTEST
```

```
                        0026        .SBTTL  "TEST   00A     RTMP0 Scratchpad Operational Test
                        0026  1521 ;*******************************************************************************
                        0026  1522 ;++
                        0026  1523 ;
                        0026  1524 ; FUNCTIONAL DESCRIPTION
                        0026  1525 ;
                        0026  1526 ;       This test checks that scratchpad RTMP0 is operational.
                        0026  1527 ;
                        0026  1528 ; TEST ALGORITHM
                        0026  1529 ;
                        0026  1530 ;       1. Load micro words to perform the test.
                        0026  1531 ;       2. Execute the micro sequence.
                        0026  1532 ;               1. Load 33333333 pattern into RTMP0.
                        0026  1533 ;               2. Read RTMP0 into RDM.
                        0026  1534 ;       3. Check the results.
                        0026  1535 ;
                        0026  1536 ; DATA PATTERNS
                        0026  1537 ;
                        0026  1538 ;       The following pattern is used to test RTEMP0.
                        0026  1539 ;
                        0026  1540 ;               33333333
                        0026  1541 ;
                        0026  1542 ; LOGIC DESCRIPTION
                        0026  1543 ;
                        0026  1544 ;       This test checks the scratchpad write (ALK Gate Array) and
                        0026  1545 ;       scratchpad read (SPA Gate Array) by loading and reading a
                        0026  1546 ;       33333333 pattern thru scratchpad RTMP0.
                        0026  1547 ;
                        0026  1548 ; ASSUMPTIONS
                        0026  1549 ;
                        0026  1550 ;       This test requires no signals from any other CPU modules but
                        0026  1551 ;       assumes an operational WBUS AND RBUS.
                        0026  1552 ;
                        0026  1553 ; ERROR DESCRIPTION
                        0026  1554 ;
                        0026  1555 ;       Data expected:33333333
                        0026  1556 ;       Data received:
                        0026  1557 ;
                        0026  1558 ;--
                        0026  1559 ;*******************************************************************************
                        0026  1560 ;///////////////////////////////////////////////////////////////////////////////
                        0026  1561
                        0026  1562         INITIALIZE                              ;DCLO low,Init scratchpads
                        0028  1563         LOADREG WDREG,2$,,Long                  ;Load RDM D register
                        0030  1564         LOADDCS 1$,,0,4                         ;Load the test micro instrs.
                        0037  1565         ERRLOOP                                 ;Set the error loop
                        0039  1566         FETCH   0                               ;Fetch the first micro word
                        003E  1567         BRSTCLK 3                               ;Clock should stop location 3
                        0042  1568         CMPREG  WHREG,2$,,Long                  ;Check the received data
                        004B  1569         IFERROR ,<<ALP><SPA><SRM>>              ;Continue if no error
                        0053  1570                                                 ;else call out ALP,SPA,SRM
                        0053  1571         SKIP
                        005A  1572
                        005A  1573 BEGIN_TEST_DATA
                        005A
```

```
                                  005A        ;-----------------------------------------------------------------------
                                  005A  1574  ;+
                                  005A  1575  ; The following micro words are used to write and read the RTMP0
                                  005A  1576  ; scrarchpad.
                                  005A  1577  ;-
                                  005A  1578
                                  005A  1579  1$:
U 00, 7700,C800,0364,0300,0470,1801  005A  1580  ;% 00   NOP                                  ;NOP Micro instruction
U 01, 5700,8840,0364,0300,0470,1802  0065  1584  ;% 01   R[0]_RDM                             ;Load test pattern
U 02, 6700,4800,5BE4,0300,0470,1803  0070  1588  ;% 02   RDM_R[0]                             ;Read test pattern
U 03, 7300,C800,0364,0300,0470,1804  007B  1592  ;% 03   NOP,STOP.CLOCK                       ;Stop clock and check results
                                  0086  1596
                          33333333  0086  1597  2$:     .Long   ^x33333333                    ;Expected received data
                                  008A  1598
                                  008A  1599  END_TEST_DATA
                                  008A
                                  008A        ;-----------------------------------------------------------------------
                                  008A  1600
                                  008A  1601  ENDTEST
```

```
                        0025        .SBTTL  "TEST   00B     IPR0 Scratchpad Operational Test
                        0025  1603 ;*******************************************************************************
                        0025  1604 ;++
                        0025  1605 ;
                        0025  1606 ; FUNCTIONAL DESCRIPTION
                        0025  1607 ;
                        0025  1608 ;       This test checks that scratchpad IPR0 is operational.
                        0025  1609 ;
                        0025  1610 ; TEST ALGORITHM
                        0025  1611 ;
                        0025  1612 ;       1. Load micro words to perform the test.
                        0025  1613 ;       2. Execute micro sequence.
                        0025  1614 ;               1. Load 33333333 pattern into IPR0.
                        0025  1615 ;               2. Read IPR0 into RDM.
                        0025  1616 ;       3. Check the results.
                        0025  1617 ;
                        0025  1618 ; DATA PATTERNS
                        0025  1619 ;
                        0025  1620 ;       The following pattern is used to test IPR0 Scratchpad:
                        0025  1621 ;
                        0025  1622 ;               33333333
                        0025  1623 ;
                        0025  1624 ; LOGIC DESCRIPTION
                        0025  1625 ;
                        0025  1626 ;       Thie test checks the scratchpad write (ALK Gate Array) and
                        0025  1627 ;       scratchpad read (SPA Gate Array) by loading and reading a
                        0025  1628 ;       33333333 pattern thru scratchpad IPR0.
                        0025  1629 ;
                        0025  1630 ; ASSUMPTIONS
                        0025  1631 ;
                        0025  1632 ;       This test requires no signals from any other CPU module but
                        0025  1633 ;       assumes an operational WBUS and RBUS.
                        0025  1634 ;
                        0025  1635 ; ERROR DESCRIPTION
                        0025  1636 ;
                        0025  1637 ;       Data expected:33333333
                        0025  1638 ;       Data received:
                        0025  1639 ;
                        0025  1640 ;--
                        0025  1641 ;*******************************************************************************
                        0025  1642 ;///////////////////////////////////////////////////////////////////////////////
                        0025  1643
                        0025  1644         INITIALIZE                          ;DCLO low,Initscratchpads
                        0027  1645         LOADREG WDREG,2$,,Long              ;Load RDM D register
                        002F  1646         LOADDCS 1$,,0,4                     ;Load the test micro instrs.
                        0036  1647         ERRLOOP                             ;Set the error loop
                        0038  1648         FETCH   0                           ;Fetch the first micro word
                        003D  1649         BRSTCLK 3                           ;Clock should stop location 3
                        0041  1650         CMPREG  WHREG,2$,,Long              ;Check the received data
                        004A  1651         IFERROR ,<<ALP>,<SPA>>              ;Continue if no error
                        0051  1652                                             ;else call out ALP,SPA
                        0051  1653         SKIP                                ;End of test
                        0058  1654
                        0058  1655 BEGIN_TEST_DATA
                        0058
```

```
                                  0058        ;-----------------------------------------------------------------------
                                  0058  1656  ;+
                                  0058  1657  ; The following words are used to write and read the IPR0
                                  0058  1658  ; scratchpad.
                                  0058  1659  ;-
                                  0058  1660
                                  0058  1661  1$:
U 00, 7700,C800,0364,0300,0470,1801  0058  1662  ;% 00   NOP                                ;NOP Micro instruction
U 01, 5700,C840,0364,0380,0470,1802  0063  1666  ;% 01   R[20]_RDM                          ;Load test pattern
U 02, 6700,0800,5BE4,0380,0470,1803  006E  1670  ;% 02   RDM_R[20]                          ;Read test pattern
U 03, 7300,C800,0364,0300,0470,1804  0079  1674  ;% 03   NOP,STOP.CLOCK                     ;Stop clock and check results
                                  0084  1678
                          33333333  0084  1679  2$:     .Long   ^x33333333                 ;Expected received data
                                  0088  1680
                                  0088  1681  END_TEST_DATA
                                  0088
                                  0088        ;-----------------------------------------------------------------------
                                  0088  1682
                                  0088  1683  ENDTEST
```

```
0031          .SBTTL  "TEST   00C     Q Register Data Integrity and Data Path Test
0031  1685 ;*******************************************************************************
0031  1686 ;++
0031  1687 ;
0031  1688 ; FUNCTIONAL DESCRIPTION
0031  1689 ;
0031  1690 ;       This test checks the Q Register and the Q Register-to-Wbus
0031  1691 ;       data path for data integrity by loading and reading six
0031  1692 ;       patterns from the LONLIT register.
0031  1693 ;
0031  1694 ; TEST ALGORITHM
0031  1695 ;
0031  1696 ;       1. Load the micro words to perform the test.
0031  1697 ;       2. Setup a loop of six iterations.
0031  1698 ;       3. Select and load the data pattern into LONLIT field
0031  1699 ;               of the micro word.
0031  1700 ;       4. Setup another loop of three iterations.
0031  1701 ;       5. Select and load ALPCTL field into micro word.
0031  1702 ;       6. Execute the micro sequence.
0031  1703 ;               1. Load LONLIT register with data pattern.
0031  1704 ;               2. Transfer pattern to Q register.
0031  1705 ;               3. Read Q register into RDM.
0031  1706 ;       7. Check the results.
0031  1707 ;       8. Checked the three data paths?If not go to step five.
0031  1708 ;       9. Loaded the six data patterns?If not go to step three.
0031  1709 ;
0031  1710 ; DATA PATTERNS
0031  1711 ;
0031  1712 ;       One of the following patterns is used each time thru the loop:
0031  1713 ;
0031  1714 ;               AAAAAAAA
0031  1715 ;               55555555
0031  1716 ;               33333333
0031  1717 ;               0F0F0F0F
0031  1718 ;               00FF00FF
0031  1719 ;               0000FFFF
0031  1720 ;
0031  1721 ; ALP Control Field Patterns
0031  1722 ;
0031  1723 ;               WX_Q.Q_D
0031  1724 ;               WX_D_Q.Q_D
0031  1725 ;               WX_D_Q.Q_M
0031  1726 ;
0031  1727 ; LOGIC DESCRIPTION
0031  1728 ;
0031  1729 ;       This test checks the Q register (which is internal to the ALP
0031  1730 ;       Gate Array) and attendent ALP Gate Array'S mux's for open or
0031  1731 ;       shorted data paths,proper ALK Gate Array and ALP Gate Array
0031  1732 ;       data path decode and clocks from the SAC Gate Array.
0031  1733 ;
0031  1734 ; ASSUMPTIONS
0031  1735 ;
0031  1736 ;       This test requires no signals from any other CPU modules but
0031  1737 ;       assumes an operational RBUS and WBUS.
0031  1738 ;
```

```
                        0031  1739 ; ERROR DESCRIPTION
                        0031  1740 ;
                        0031  1741 ;       Data Expected: AAAAAAAA          ;First Loop
                        0031  1742 ;                    : 55555555          ;Second Loop
                        0031  1743 ;                    : 33333333          ;Third Loop
                        0031  1744 ;                    : 0F0F0F0F          ;Fourth Loop
                        0031  1745 ;                    : 00FF00FF          ;Fifth Loop
                        0031  1746 ;                    : 0000FFFF          ;Sixth Loop
                        0031  1747 ;       Data Received:
                        0031  1748 ;       Loop Count I:                    ;Data Patterns
                        0031  1749 ;       Loop Count J:                    ;ALP Control Field
                        0031  1750 ;--
                        0031  1751 ;*****************************************************************************
                        0031  1752 ;/////////////////////////////////////////////////////////////////////////////
                        0031  1753 ;+
                        0031  1754 ; Select and load test pattern into LONLIT Register,transfer to Q
                        0031  1755 ;  Register,read into RDM,select new data path and read Q Register
                        0031  1756 ;    into RDM again.
                        0031  1757 ;-
                        0031  1758         INITIALIZE                       ;DCLO low,Initscratchpads
                        0033  1759         LOOP    I,1,6                    ;Setup loop to load LONLIT
                        003A  1760         SA01PAT I,2$,Longlit,4$          ;Generate test patterns and
                        0042  1761         LOOP    J,1,3                    ;Setup loop to load ALPCTL
                        0049  1762         LDFIELD 5$,J,Word,3$,48,57       ;Load ALPCTL field
                        005A  1763         LOADDCS 1$,,0,5                  ;Load the test micro instrs
                        0061  1764                                          ;load into LONLIT field
                        0061  1765         ERRLOOP                          ;Set the error loop
                        0063  1766         FETCH   0                        ;Fetch the first micro instr
                        0068  1767         BRSTCLK 4                        ;Clock stop location 4
                        006C  1768         CMPREG  WHREG,4$,,Long           ;Check the received data
                        0075  1769         IFERROR ,<<ALP>,<ALK>>           ;Continue if no error
                        007C  1770                                          ;else call out ALP,ALK
                        007C  1771         ENDLOOP J                        ;Continue with loop
                        007F  1772         ENDLOOP I                        ;Continue with loop
                        0082  1773         SKIP                             ;End of test
                        0089  1774
                        0089  1775 BEGIN_TEST_DATA
                        0089
                        0089          ;-------------------------------------------------------------------
                        0089  1776 ;+
                        0089  1777 ; The following micro words are used to write and read the Q register.
                        0089  1778 ;-
                        0089  1779
                        0089  1780 1$:
U 00, 7700,C800,0364,0300,0470,1801     0089  1781 ;% 00    NOP                          ;NOP Micro instruction
                        0094  1785 2$:
U 01, 7700,7800,7FFF,FFFF,8470,1802     0094  1786 ;% 01    LONLIT_[0]                   ;LONLIT gets test pattern
U 02, 7700,4800,03C7,03D4,0470,1803     009F  1790 ;% 02    Q_LONLIT                     ;Load Q Register
                        00AA  1794 3$:
U 03, 6700,8800,02C7,0300,0470,1804     00AA  1795 ;% 03    RDM_Q_Q_D                    ;ALPCTL/5$
U 04, 7300,4800,0364,0300,0470,1805     00B5  1799 ;% 04    NOP,STOP.CLOCK               ;Stop clock,check results
                        00C0  1803
                    00000000    00C0  1804 4$:      .Long                        ;Expected data goes here
                        00C4  1805
                        02C7    00C4  1806 5$:      .Word   ^x2C7
```

```
                        02D7  00C6  1807          .Word   ^x2D7
                        00D7  00C8  1808          .Word   ^xD7
                              00CA  1809
                              00CA  1810 END_TEST_DATA
                              00CA
                              00CA        ;-----------------------------------------------------------------------
                              00CA  1811
                              00CA  1812 ENDTEST
```

```
0019        .SBTTL  "TEST   00D     D Register Data Test
0019  1814 ;*******************************************************************************
0019  1815 ;++
0019  1816 ;
0019  1817 ; FUNCTIONAL DESCRIPTION
0019  1818 ;
0019  1819 ;       This test checks the D Register for data integrity by selecting and
0019  1820 ;       loading one of three patterns into the LONLIT Register,from the
0019  1821 ;       LONLIT register into the D Register,loading the Q Register from the
0019  1822 ;       D Register and reading data into the RDM from the Q Register.
0019  1823 ;
0019  1824 ; TEST ALGORITHM
0019  1825 ;
0019  1826 ;       1. Load the micro words to perform the test.
0019  1827 ;       2. Setup loop of three iterations.
0019  1828 ;       3. Select and load data pattern.
0019  1829 ;       4. Execute the micro sequence.
0019  1830 ;               1. Load LONLIT register with selected pattern.
0019  1831 ;               2. Load D REGISTER from LONLIT register.
0019  1832 ;               3. Load Q register from D REGISTER.
0019  1833 ;               4. Read Q register into RDM.
0019  1834 ;       5. Check the results.
0019  1835 ;       6. Looped three times?If not go to step three.
0019  1836 ;
0019  1837 ; DATA PATTERNS
0019  1838 ;
0019  1839 ;       The following pattern are used to test the "D" Register:
0019  1840 ;
0019  1841 ;               AAAAAAAA
0019  1842 ;               55555555
0019  1843 ;               33333333
0019  1844 ;
0019  1845 ; ALP Control Field
0019  1846 ;
0019  1847 ;         ALPCTL/WX_D_R.Q_D
0019  1848 ;
0019  1849 ; LOGIC DESCRIPTION
0019  1850 ;
0019  1851 ;       This test checks the D REGISTER (which is internal to the ALP
0019  1852 ;       Gate Array) and attendent ALP Gate Array's Mux's for open or
0019  1853 ;       shorted data paths and proper ALK Gate Array and ALP Gate
0019  1854 ;       Array data path decode.
0019  1855 ;
0019  1856 ; ASSUMPTIONS
0019  1857 ;
0019  1858 ;       This test requires no signals from the other CPU modules but
0019  1859 ;       assumes an operational RBUS and WBUS.
0019  1860 ;
0019  1861 ; ERROR DESCRIPTION
0019  1862 ;
0019  1863 ;       Data Expected: Same as last written
0019  1864 ;       Data Received:
0019  1865 ;       Loop Count:
0019  1866 ;
0019  1867 ;--
```

```
                     0019  1868 ;*****************************************************************************
                     0019  1869 ;/////////////////////////////////////////////////////////////////////////////
                     0019  1870 ;+
                     0019  1871 ; Select and load patterns into D register,transfer to Q Register and
                     0019  1872 ;  read into RDM.
                     0019  1873 ;-
                     0019  1874
                     0019  1875         INITIALIZE                          ;DCLO Low,Init scratchpads
                     001B  1876         LOOP    I,1,3                       ;Init loop count
                     0022  1877         SA01PAT I,2$,Longlit,3$             ;Select and load test pattern
                     002A  1878         LOADDCS 1$,,0,6                     ;Load test micro instructions
                     0031  1879         ERRLOOP                             ;Set the error loop
                     0033  1880         FETCH   0                           ;Fetch the first micro instr.
                     0038  1881         BRSTCLK 5                           ;Clock stop location 5.
                     003C  1882         CMPREG  WHREG,3$,,Long              ;Check the received data.
                     0045  1883         IFERROR ,<<ALP>,<ALK>>              ;Continue if no error
                     004C  1884                                             ;else call out ALP,ALK
                     004C  1885         ENDLOOP I                           ;Continue with loop
                     004F  1886         SKIP                                ;End of test
                     0056  1887
                     0056  1888 BEGIN_TEST_DATA
                     0056
                     0056       ;-------------------------------------------------------------------
                     0056  1889 ;+
                     0056  1890 ; The following micro words write and read the D Register.
                     0056  1891 ;-
                     0056  1892
                     0056  1893 1$:
U 00, 7700,C800,0364,0300,0470,1801  0056  1894 ;% 00    NOP                 ;NOP Micrpo instruction
                     0061  1898 2$:
U 01, 7700,7800,7FFF,FFFF,8470,1802  0061  1899 ;% 01    LONLIT_[0]          ;LONLIT/Test pattern
U 02, 7700,0800,0257,03D4,0470,1803  006C  1903 ;% 02    D_LONLIT
U 03, 7700,C800,0247,0300,0470,1804  0077  1907 ;% 03    Q_D WX_R
U 04, 6700,0800,02C7,0300,0470,1805  0082  1911 ;% 04    RDM_Q Q_D           ;Check test pattern
U 05, 7300,4800,0364,0300,0470,1806  008D  1915 ;% 05    NOP,STOP.CLOCK      ;Stop clock,check results
                     0098  1919
                  00000000  0098  1920 3$:     .Long                       ;Expected data goes here
                     009C  1921
                     009C  1922 END_TEST_DATA
                     009C
                     009C       ;-------------------------------------------------------------------
                     009C  1923
                     009C  1924 ENDTEST
```

```
0024          .SBTTL  "TEST   00E     RTEMP Scratchpad Data Integrity
0024  1926 ;*******************************************************************************
0024  1927 ;++
0024  1928 ;
0024  1929 ; FUNCTIONAL DESCRIPTION
0024  1930 ;
0024  1931 ;       This test checks the data integrity of the RTEMP registers by
0024  1932 ;       loading and reading six patterns.
0024  1933 ;
0024  1934 ; TEST ALGORITHM
0024  1935 ;
0024  1936 ;       1. Load micro words to perform the test.
0024  1937 ;       2. Setup a loop of six iterations.
0024  1938 ;       3. Select and load a data pattern into the LONLIT field of
0024  1939 ;               the micro word.
0024  1940 ;       4. Execute the micro sequence.
0024  1941 ;               1. Load test pattern into LONLIT Register.
0024  1942 ;               2. Load pattern into Q Register.
0024  1943 ;               3. Load pattern into R[0] from Q Register.
0024  1944 ;               4. Read pattern into RDM.
0024  1945 ;       5. Check the results.
0024  1946 ;       6. Looped six times? if not,go to step three.
0024  1947 ;
0024  1948 ; DATA PATTERNS
0024  1949 ;
0024  1950 ;       One of the following patterns is used each time thru the loop:
0024  1951 ;
0024  1952 ;               AAAAAAAA
0024  1953 ;               55555555
0024  1954 ;               33333333
0024  1955 ;               0F0F0F0F
0024  1956 ;               00FF00FF
0024  1957 ;               0000FFFF
0024  1958 ;
0024  1959 ; LOGIC DESCRIPTION
0024  1960 ;
0024  1961 ;       This test checks that the RTEMP scratchpads' data lines are not SA0,
0024  1962 ;       SA1 or shorted shorted together.
0024  1963 ;
0024  1964 ; ASSUMPTIONS
0024  1965 ;
0024  1966 ;       This test requires no signals from other CPU modules but assumes an
0024  1967 ;       operational WBUS and RBUS.
0024  1968 ;
0024  1969 ; ERROR DESCRIPTION
0024  1970 ;
0024  1971 ;       Data expected:Same as last pattern loaded.
0024  1972 ;       Data received:
0024  1973 ;       Loop Count:
0024  1974 ;
0024  1975 ;--
0024  1976 ;*******************************************************************************
0024  1977 ;///////////////////////////////////////////////////////////////////////////////
0024  1978
0024  1979          INITIALIZE                          ;DCLO low,Init scratchpads
```

```
                        0026  1980          LOOP    I,1,6                   ;Initialize loop count
                        002D  1981          SA01PAT I,2$,Longlit,3$         ;Select a test pattern
                        0035  1982          LOADDCS 1$,,0,6                 ;Load the test micro instrs.
                        003C  1983          ERRLOOP                         ;Set the error loop
                        003E  1984          FETCH   0                       ;Load first micro word
                        0043  1985          BRSTCLK 5                       ;Clock should stop location 5
                        0047  1986          CMPREG  WHREG,3$,,Long          ;Check the received data
                        0050  1987          IFERROR ,<<ALP><SRM>>           ;If error call out ALP and SRM
                        0057  1988          ENDLOOP I                       ;Continue with loop
                        005A  1989          SKIP                            ;End of test
                        0061  1990
                        0061  1991  BEGIN_TEST_DATA
                        0061
                        0061          ;-----------------------------------------------------------------------
                        0061  1992  ;+
                        0061  1993  ; The following micro words are used to write and read the R[0] scratchpad.
                        0061  1994  ;-
                        0061  1995
                        0061  1996  1$:
U 00, 7700,C800,0364,0300,0470,1801  0061  1997  ;% 00   NOP                     ;NOP Micro instruction
                        006C  2001  2$:
U 01, 7700,7800,7FFF,FFFF,8470,1802  006C  2002  ;% 01   LONLIT_[0]             ;Put pattern into LONLIT Register
U 02, 7700,4800,03C7,03D4,0470,1803  0077  2006  ;% 02   Q_LONLIT              ;Transfer pattern to Q Register
U 03, 7700,C840,02C7,0300,0470,1804  0082  2010  ;% 03   R[0]_Q Q_D            ;Put pattern into RTEMP0
U 04, 6700,4800,5BE4,0300,0470,1805  008D  2014  ;% 04   RDM_R[0]             ;Read test pattern
U 05, 7300,4800,0364,0300,0470,1806  0098  2018  ;% 05   NOP,STOP.CLOCK       ;Stop clock and check results
                        00A3  2022
                00000000  00A3  2023  3$:     .Long                          ;Test pattern goes here
                        00A7  2024
                        00A7  2025  END_TEST_DATA
                        00A7
                        00A7          ;-----------------------------------------------------------------------
                        00A7  2026
                        00A7  2027  ENDTEST
```

```
                        0022        .SBTTL  "TEST   00F     GPR Scratchpad Data Integrity
                        0022  2029 ;*****************************************************************************
                        0022  2030 ;++
                        0022  2031 ;
                        0022  2032 ; FUNCTIONAL DESCRIPTION
                        0022  2033 ;
                        0022  2034 ;       This test checks the data integrity of the GPR registers by
                        0022  2035 ;       loading and reading six patterns.
                        0022  2036 ;
                        0022  2037 ; TEST ALGORITHM
                        0022  2038 ;
                        0022  2039 ;       1. Load micro words to perform the test.
                        0022  2040 ;       2. Setup a loop of six iterations.
                        0022  2041 ;       3. Select and load a data pattern into the LONLIT field of
                        0022  2042 ;               the micro word.
                        0022  2043 ;       4. Execute the micro sequence.
                        0022  2044 ;               1. Load test pattern into LONLIT Register.
                        0022  2045 ;               2. Load pattern into Q Register.
                        0022  2046 ;               3. Load pattern into R[10] from Q Register.
                        0022  2047 ;               4. Read pattern into RDM.
                        0022  2048 ;       5. Check the results.
                        0022  2049 ;       6. Looped six times? if not,go to step three.
                        0022  2050 ;
                        0022  2051 ; DATA PATTERNS
                        0022  2052 ;
                        0022  2053 ;       One of the following patterns is used each time thru the loop:
                        0022  2054 ;
                        0022  2055 ;               AAAAAAAA
                        0022  2056 ;               55555555
                        0022  2057 ;               33333333
                        0022  2058 ;               0F0F0F0F
                        0022  2059 ;               00FF00FF
                        0022  2060 ;               0000FFFF
                        0022  2061 ;
                        0022  2062 ; LOGIC DESCRIPTION
                        0022  2063 ;
                        0022  2064 ;       This test checks that the GPR scratchpads' data lines are not SA0,
                        0022  2065 ;       SA1 or shorted shorted together.
                        0022  2066 ;
                        0022  2067 ; ASSUMPTIONS
                        0022  2068 ;
                        0022  2069 ;       This test requires no signals from other CPU modules but assumes an
                        0022  2070 ;       operational WBUS and RBUS.
                        0022  2071 ;
                        0022  2072 ; ERROR DESCRIPTION
                        0022  2073 ;
                        0022  2074 ;       Data expected:Same as last pattern loaded.
                        0022  2075 ;       Data received:
                        0022  2076 ;       Loop Count:
                        0022  2077 ;
                        0022  2078 ;--
                        0022  2079 ;*****************************************************************************
                        0022  2080 ;/////////////////////////////////////////////////////////////////////////////
                        0022  2081
                        0022  2082         INITIALIZE                      ;DCLO low,Init scratchpads
```

```
                        0024  2083          LOOP    I,1,6                   ;Initialize loop count
                        002B  2084          SA01PAT I,2$,Longlit,3$         ;Select a test pattern
                        0033  2085          LOADDCS 1$,,0,6                 ;Load the test micro instrs.
                        003A  2086          ERRLOOP                         ;Set the error loop
                        003C  2087          FETCH   0                       ;Load first micro word
                        0041  2088          BRSTCLK 5                       ;Clock should stop location 5
                        0045  2089          CMPREG  WHREG,3$,,Long          ;Check the received data
                        004E  2090          IFERROR ,<<ALP>>                ;Continue if no error else call out ALP
                        0054  2091          ENDLOOP I                       ;Continue with loop
                        0057  2092          SKIP                            ;End of test
                        005E  2093
                        005E  2094 BEGIN_TEST_DATA
                        005E
                        005E          ;-----------------------------------------------------------------------
                        005E  2095 ;+
                        005E  2096 ; The following micro words are used to write and read the
                        005E  2097 ;  R[10] scratchpad.
                        005E  2098 ;-
                        005E  2099
                        005E  2100 1$:
U 00, 7700,C800,0364,0300,0470,1801     005E  2101 ;% 00    NOP                             ;NOP Micro instruction
                        0069  2105 2$:
U 01, 7700,7800,7FFF,FFFF,8470,1802     0069  2106 ;% 01    LONLIT_[0]                      ;Test pattern goes here
U 02, 7700,4800,03C7,03D4,0470,1803     0074  2110 ;% 02    Q_LONLIT                        ;Load test pattern into Q Reg.
U 03, 7700,8840,02C7,0340,0470,1804     007F  2114 ;% 03    R[10]_Q Q_D
U 04, 6700,0800,5BE4,0340,0470,1805     008A  2118 ;% 04    RDM_R[10]
U 05, 7300,4800,0364,0300,0470,1806     0095  2122 ;% 05    NOP,STOP.CLOCK                  ;Stop clock and check results
                        00A0  2126
                00000000  00A0  2127 3$:      .Long                           ;Test pattern goes here
                        00A4  2128
                        00A4  2129 END_TEST_DATA
                        00A4
                        00A4          ;-----------------------------------------------------------------------
                        00A4  2130
                        00A4  2131 ENDTEST
```

```
                        0022        .SBTTL  "TEST   010     IPR Scratchpad Data Integrity
                        0022  2133 ;*******************************************************************************
                        0022  2134 ;++
                        0022  2135 ;
                        0022  2136 ; FUNCTIONAL DESCRIPTION
                        0022  2137 ;
                        0022  2138 ;       This test checks the data integrity of the IPR registers by
                        0022  2139 ;       loading and reading six patterns.
                        0022  2140 ;
                        0022  2141 ; TEST ALGORITHM
                        0022  2142 ;
                        0022  2143 ;       1. Load micro words to perform the test.
                        0022  2144 ;       2. Setup a loop of six iterations.
                        0022  2145 ;       3. Select and load a data pattern into the LONLIT field of
                        0022  2146 ;               the micro word.
                        0022  2147 ;       4. Execute the micro sequence.
                        0022  2148 ;               1. Load test pattern into LONLIT Register.
                        0022  2149 ;               2. Load pattern into Q Register.
                        0022  2150 ;               3. Load pattern into R[20] from Q Register.
                        0022  2151 ;               4. Read pattern into RDM.
                        0022  2152 ;       5. Check the results.
                        0022  2153 ;       6. Looped six times? if not,go to step three.
                        0022  2154 ;
                        0022  2155 ; DATA PATTERNS
                        0022  2156 ;
                        0022  2157 ;       One of the following patterns is used each time thru the loop:
                        0022  2158 ;
                        0022  2159 ;               AAAAAAAA
                        0022  2160 ;               55555555
                        0022  2161 ;               33333333
                        0022  2162 ;               0F0F0F0F
                        0022  2163 ;               00FF00FF
                        0022  2164 ;               0000FFFF
                        0022  2165 ;
                        0022  2166 ; LOGIC DESCRIPTION
                        0022  2167 ;
                        0022  2168 ;       This test checks that the IPR scratchpads' data lines are not SA0,
                        0022  2169 ;       SA1 or shorted shorted together.
                        0022  2170 ;
                        0022  2171 ; ASSUMPTIONS
                        0022  2172 ;
                        0022  2173 ;       This test requires no signals from other CPU modules but assumes an
                        0022  2174 ;       operational WBUS and RBUS.
                        0022  2175 ;
                        0022  2176 ; ERROR DESCRIPTION
                        0022  2177 ;
                        0022  2178 ;       Data expected:Same as last pattern loaded.
                        0022  2179 ;       Data received:
                        0022  2180 ;       Loop Count:
                        0022  2181 ;
                        0022  2182 ;--
                        0022  2183 ;*******************************************************************************
                        0022  2184 ;///////////////////////////////////////////////////////////////////////////////
                        0022  2185
                        0022  2186         INITIALIZE                          ;DCLO low,Init scratchpads    .
```

```
                        0024  2187          LOOP    I,1,6                   ;Initialize loop count
                        002B  2188          SA01PAT I,2$,Longlit,3$         ;Select a test pattern
                        0033  2189          LOADDCS 1$,,0,6                 ;Load the test micro instrs.
                        003A  2190          ERRLOOP                         ;Set the error loop
                        003C  2191          FETCH   0                       ;Load first micro word
                        0041  2192          BRSTCLK 5                       ;Clock should stop location 5
                        0045  2193          CMPREG  WHREG,3$,,Long          ;Check the received data
                        004E  2194          IFERROR ,<<ALP>>                ;Continue if no error else call out ALP
                        0054  2195          ENDLOOP I                       ;Continue with loop
                        0057  2196          SKIP                            ;End of test
                        005E  2197
                        005E  2198 BEGIN_TEST_DATA
                        005E
                        005E              ;-----------------------------------------------------------------------
                        005E  2199 ;+
                        005E  2200 ; The following micro words are used to write and read the
                        005E  2201 ;  R[20] scratchpad.
                        005E  2202 ;-
                        005E  2203
                        005E  2204 1$:
U 00, 7700,C800,0364,0300,0470,1801     005E  2205 ;% 00    NOP                             ;NOP Micro instruction
                        0069  2209 2$:
U 01, 7700,7800,7FFF,FFFF,8470,1802     0069  2210 ;% 01    LONLIT_[0]                      ;Test pattern goes here
U 02, 7700,4800,03C7,03D4,0470,1803     0074  2214 ;% 02    Q_LONLIT                        ;Load test pattern into Q Reg.
U 03, 7700,8840,02C7,0380,0470,1804     007F  2218 ;% 03    R[20]_Q Q_D
U 04, 6700,0800,5BE4,0380,0470,1805     008A  2222 ;% 04    RDM_R[20]                       ;Read test pattern
U 05, 7300,4800,0364,0300,0470,1806     0095  2226 ;% 05    NOP,STOP.CLOCK                  ;Stop clock and check results
                        00A0  2230
                00000000  00A0  2231 3$:      .Long                           ;Test pattern goes here
                        00A4  2232
                        00A4  2233 END_TEST_DATA
                        00A4
                        00A4          ;-----------------------------------------------------------------------
                        00A4  2234
                        00A4  2235 ENDTEST
```

```
                        0022        .SBTTL  "TEST   011     R Scratchpad Chip Select Test
                        0022  2237  ;*******************************************************************************
                        0022  2238  ;++
                        0022  2239  ;
                        0022  2240  ; FUNCTIONAL DESCRIPTION
                        0022  2241  ;
                        0022  2242  ;       This test checks that only one R scratchpad register file is written
                        0022  2243  ;       at a time.A register file,in the following descriptions,is location
                        0022  2244  ;       0 of the RTEMP,GPR or IPR register sets.
                        0022  2245  ;
                        0022  2246  ; TEST ALGORITHM
                        0022  2247  ;
                        0022  2248  ;       1. Load micro words to perform the test.
                        0022  2249  ;       2. Setup loop of four iterations.
                        0022  2250  ;       3. Load fields with selected RSRC values.
                        0022  2251  ;       4. Execute the micro sequence.
                        0022  2252  ;               1. Write AAAAAAAA into one register file.
                        0022  2253  ;               2. Write 55555555 into other two register files.
                        0022  2254  ;               3. Read AAAAAAAA from first register file.
                        0022  2255  ;       5. Check the results.
                        0022  2256  ;       6. Looped four times?If not go to step three.
                        0022  2257  ;
                        0022  2258  ; DATA PATTERNS
                        0022  2259  ;
                        0022  2260  ;               AAAAAAAA
                        0022  2261  ;               55555555
                        0022  2262  ;
                        0022  2263  ;
                        0022  2264  ; RSRC VALUES
                        0022  2265  ;
                        0022  2266  ;                   00
                        0022  2267  ;                   10
                        0022  2268  ;                   20
                        0022  2269  ;
                        0022  2270  ;
                        0022  2271  ; LOGIC DESCRIPTION
                        0022  2272  ;
                        0022  2273  ;       This test checks that the scratchpad chip selects are not
                        0022  2274  ;       open nor shorted together.
                        0022  2275  ;
                        0022  2276  ; ASSUMPTIONS
                        0022  2277  ;
                        0022  2278  ;       This test requires no signals from other CPU modules.It
                        0022  2279  ;       assumes an operational WBUS and RBUS.
                        0022  2280  ;
                        0022  2281  ; ERROR DESCRIPTION
                        0022  2282  ;
                        0022  2283  ;       Data expected:AAAAAAAA
                        0022  2284  ;       Data received:
                        0022  2285  ;       Loop Count:
                        0022  2286  ;
                        0022  2287  ;--
                        0022  2288  ;*******************************************************************************
                        0022  2289  ;///////////////////////////////////////////////////////////////////////////////
                        0022  2290  ;+
```

```
                        0022  2291 ; Load AAAAAAAA pattern into first register,the 55555555 pattern
                        0022  2292 ; into second and third registers and read AAAAAAAA from first reg.
                        0022  2293 ;-
                        0022  2294         INITIALIZE                          ;DCLO Low,Init scratchpads
                        0024  2295         LOOP    I,1,4                       ;Set loop count
                        002B  2296         LDFIELD 6$,I,Byte,2$,34,39          ;RSRC/6$
                        003C  2297         LDFIELD 7$,I,Byte,3$,34,39          ;RSRC/7$
                        004D  2298         LDFIELD 8$,I,Byte,4$,34,39          ;RSRC/8$
                        005E  2299         LDFIELD 6$,I,Byte,5$,34,39          ;RSRC/6$
                        006F  2300         LOADDCS 1$,,0,<^xB>                 ;Load the modified micro-instructions
                        0076  2301         ERRLOOP                             ;Set the error loop
                        0078  2302         FETCH   0                           ;Get first micro word
                        007D  2303         BRSTCLK <^xA>                       ;Clock should stop location <x>A
                        0081  2304         CMPREG  WHREG,9$,,Long              ;Check the received data
                        008A  2305         IFERROR ,<<SPA>>                    ;Continue if no error else call out SPA
                        0090  2306         ENDLOOP I                           ;Continue with loop
                        0093  2307         SKIP                                ;End of test
                        009A  2308
                        009A  2309 BEGIN_TEST_DATA
                        009A
                        009A       ;-----------------------------------------------------------------------
                        009A  2310 ;+
                        009A  2311 ; The following micro words are used to write and read the R
                        009A  2312 ;   scratchpads.
                        009A  2313 ;-
                        009A  2314 1$:
U 00, 7700,C800,0364,0300,0470,1801  009A  2315 ;% 00   NOP                         ;NOP Micro instruction
U 01, 7700,7800,2AAA,AAAA,8470,1802  00A5  2319 ;% 01   LONLIT_[0AAAAAAAA]         ;Load LONLIT register
U 02, 7700,4800,03C7,03D4,0470,1803  00B0  2323 ;% 02   Q_LONLIT                   ;Load Q register
                                     00BB  2327 2$:
U 03, 7700,C840,02C7,0300,0470,1804  00BB  2328 ;% 03   R[0]_Q Q_D                ;Load R Scratchpad
U 04, 7700,3800,5555,5555,0470,1805  00C6  2332 ;% 04   LONLIT_[55555555]        ;Load LONLIT register
U 05, 7700,0800,0257,03D4,0470,1806  00D1  2336 ;% 05   D_LONLIT                  ;Load D register
U 06, 7700,C800,0247,0300,0470,1807  00DC  2340 ;% 06   Q_D WX_R                 ;Load Q Register
                                     00E7  2344 3$:
U 07, 7700,8840,02C7,0340,0470,1808  00E7  2345 ;% 07   R[10]_Q Q_D              ;Load R Scratchpad
                                     00F2  2349 4$:
U 08, 7700,0840,02C7,0380,0470,1809  00F2  2350 ;% 08   R[20]_Q Q_D              ;Load R Scratchpad
                                     00FD  2354 5$:
U 09, 6700,4800,5BE4,0300,0470,180A  00FD  2355 ;% 09   RDM_R[0]                 ;Load RDM
U 0A, 7300,C800,0364,0300,0470,180B  0108  2359 ;% 0A   NOP,STOP.CLOCK           ;Stop clock,check results
                                     0113  2363
                              00     0113  2364 6$:     .Byte   0               ;RTMP0
                              10     0114  2365        .Byte   ^X10            ;GPR0
                              20     0115  2366        .Byte   ^X20            ;IPR0
                              00     0116  2367        .Byte   0               ;RTMP0
                                     0117  2368
                              10     0117  2369 7$:     .Byte   ^X10            ;GPR0
                              20     0118  2370        .Byte   ^X20            ;IPR0
                              00     0119  2371        .Byte   0               ;RTMP0
                              10     011A  2372        .Byte   ^X10            ;GPR0
                                     011B  2373
                              20     011B  2374 8$:     .Byte   ^X20            ;IPR0
                              00     011C  2375        .Byte   0               ;RTMP0
                              10     011D  2376        .Byte   ^X10            ;GPR0
```

```
                      20  011E  2377          .Byte   ^X20                    ;IPR0
                          011F  2378
                AAAAAAAA  011F  2379 9$:      .Long   ^xAAAAAAAA             ;Expected received data
                          0123  2380
                          0123  2381 END_TEST_DATA
                          0123
                          0123        ;-----------------------------------------------------------------------
                          0123  2382
                          0123  2383 ENDTEST
```

```
0026       .SBTTL  "TEST   012     R Scratchpad Write All Cells Test
0026  2385 ;*******************************************************************************
0026  2386 ;++
0026  2387 ;
0026  2388 ; FUNCTIONAL DESCRIPTION
0026  2389 ;
0026  2390 ;       This test consists of three subtests,one for each group of R
0026  2391 ;       Scratchpads,that writes each location with three patterns and
0026  2392 ;       reads it back into the RDM for checking.
0026  2393 ;
0026  2394 ; TEST ALGORITHM
0026  2395 ;
0026  2396 ;       1. Load the micro words to perform the test.
0026  2397 ;       2. Setup a loop of 16 iterations.
0026  2398 ;       3. Select and load a data address.
0026  2399 ;       4. Setup a loop of 3 iterations.
0026  2400 ;       5. Select and load a data pattern.
0026  2401 ;       6. Load the micro word with the pattern.
0026  2402 ;       7. Execute the micro sequence.
0026  2403 ;               1. Load pattern into LONLIT Register.
0026  2404 ;               2. Transfer pattern to Q Register.
0026  2405 ;               3. Transfer pattern to selected scratchpad.
0026  2406 ;               4. Read pattern into RDM.
0026  2407 ;       8. Check the results.
0026  2408 ;       9. Looped 3 times? if not go to step five.
0026  2409 ;       10.Looped 16 times? if not go to step three.
0026  2410 ;
0026  2411 ; DATA TABLES
0026  2412 ;
0026  2413 ;       SUBTEST 1       SUBTEST 2       SUBTEST 3
0026  2414 ;
0026  2415 ;       Reg.  Data      Reg.  Data      Reg.  Data
0026  2416 ;
0026  2417 ;       R0 <- AAAAAAAA  R10<- AAAAAAAA  R20<- AAAAAAAA
0026  2418 ;       RDM<- R0        RDM<- R10       RDM<- R20
0026  2419 ;       R0 <- 55555555  R10<- 55555555  R20<- 55555555
0026  2420 ;       RDM<- R0        RDM<- R10       RDM<- R20
0026  2421 ;       R0 <- 33333333  R10<- 33333333  R20<- 33333333
0026  2422 ;       RDM<- R0        RDM<- R10       RDM<- R20
0026  2423 ;
0026  2424 ;       R1 <- AAAAAAAA  R11<- AAAAAAAA  R21<- AAAAAAAA
0026  2425 ;          .               .               .
0026  2426 ;          .               .               .
0026  2427 ;       R0F<- 33333333  R1F<- 33333333  R2F<- 33333333
0026  2428 ;       RDM<- R0F       RDM<- R1F       RDM<- R2F
0026  2429 ;
0026  2430 ; DATA PATTERNS
0026  2431 ;
0026  2432 ;       One of the following patterns is used each time thru the loop:
0026  2433 ;
0026  2434 ;               AAAAAAAA
0026  2435 ;               55555555
0026  2436 ;               33333333
0026  2437 ;
0026  2438 ;
```

```
                        0026  2439 ; DATA ADDRESSES
                        0026  2440 ;
                        0026  2441 ;       The data addresses go from Zero to 2F(Hexadecimal) in ascending order.
                        0026  2442 ;
                        0026  2443 ; LOGIC DESCRIPTION
                        0026  2444 ;
                        0026  2445 ;       This test checks the R scratchpad ram chips.
                        0026  2446 ;
                        0026  2447 ; ASSUMPTIONS
                        0026  2448 ;
                        0026  2449 ;       This test requires no signals from any other cpu modules but
                        0026  2450 ;       assumes an operational RBUS and WBUS.
                        0026  2451 ;
                        0026  2452 ; ERROR DESCRIPTION
                        0026  2453 ;
                        0026  2454 ;       EXPECTED DATA
                        0026  2455 ;       RECEIVED DATA
                        0026  2456 ;       DATA ADDRESS (LOOP COUNT I)
                        0026  2457 ;       DATA PATTERN (LOOP COUNT J)
                        0026  2458 ;
                        0026  2459 ;--
                        0026  2460 ;*****************************************************************************
                        0026  2461
                        0026  2462         SUBTEST 1
                        0029
                        0029       ;////////////////////////////////////////////////////////////////////////////
                        0029  2463 ;+
                        0029  2464 ; Write the three data patterns into every RTEMP ram location and
                        0029  2465 ;  read them back.
                        0029  2466 ;-
                        0029  2467
                        0029  2468                                          ;Start test 000C
                        0029  2469         INITIALIZE                       ;DCLO Low,init scratchpads
                        002B  2470         LDFIELD 7$,,Byte,3$,38,39        ;Init micro words to RTMP
                        003C  2471         LDFIELD 7$,,Byte,4$,38,39        ;Init micro words to RTMP
                        004D  2472         LOOP    I,1,16                   ;Init loop count
                        0054  2473         LDFIELD 5$,I,Byte,3$,34,37       ;Select R no. for data
                        0065  2474         LDFIELD 5$,I,Byte,4$,34,37       ;Select R no. for data
                        0076  2475         LOOP    J,1,3                    ;Init loop count
                        007D  2476         SA01PAT J,2$,Longlit,6$          ;Select data pattern
                        0085  2477         LOADDCS 1$,,0,6                  ;Load test micro words
                        008C  2478         ERRLOOP                          ;Set error loop
                        008E  2479         FETCH   0                        ;Get first micro instruction
                        0093  2480         BRSTCLK 5                        ;Clock stop location six
                        0097  2481         CMPREG  WHREG,6$,,Long           ;Check the results
                        00A0  2482         IFERROR ,,<<DPM>>                ;Continue if no error else call out DPM
                        00A6  2483         ENDLOOP J                        ;Get next pattern
                        00A9  2484         ENDLOOP I                        ;Get next address
                        00AC  2485
                        00AC  2486         SUBTEST 2
                        00AF
                        00AF       ;////////////////////////////////////////////////////////////////////////////
                        00AF  2487 ;+
                        00AF  2488 ; Write the three data patterns into every GPR ram location and
                        00AF  2489 ;  read them back.
```

```
                        00AF  2490 ;-
                        00AF  2491         INITIALIZE                          ;DCLO Low,init scratchpads
                        00B1  2492         LDFIELD 8$,,Byte,3$,38,39           ;Init micro words to GPR
                        00C2  2493         LDFIELD 8$,,Byte,4$,38,39           ;Init micro words to GPR
                        00D3  2494         LOOP    I,1,16                      ;Init loop count
                        00DA  2495         LDFIELD 5$,I,Byte,3$,34,37          ;Select R no. for data
                        00EB  2496         LDFIELD 5$,I,Byte,4$,34,37          ;Select R no. for data
                        00FC  2497         LOOP    J,1,3                       ;Init loop count
                        0103  2498         SA01PAT J,2$,Longlit,6$             ;Select data pattern
                        010B  2499         LOADDCS 1$,,0,6                     ;Load test micro words
                        0112  2500         ERRLOOP                             ;Set error loop
                        0114  2501         FETCH   0                           ;Get first micro instruction
                        0119  2502         BRSTCLK 5                           ;Clock stop location six
                        011D  2503         CMPREG  WHREG,6$,,Long              ;Check the results
                        0126  2504         IFERROR ,,<<DPM>>                   ;Continue if no error else call out DPM
                        012C  2505         ENDLOOP J                           ;Get next pattern
                        012F  2506         ENDLOOP I                           ;Get next address
                        0132  2507
                        0132  2508         SUBTEST 3
                        0135
                        0135       ;////////////////////////////////////////////////////////////////////////////////
                        0135  2509 ;+
                        0135  2510 ; Write the three data patterns into every IPR ram location and
                        0135  2511 ;  read them back.
                        0135  2512 ;-
                        0135  2513         INITIALIZE                          ;DCLO Low,Init scratchpads
                        0137  2514         LDFIELD 9$,,Byte,3$,38,39           ;Init micro words to IPR
                        0148  2515         LDFIELD 9$,,Byte,4$,38,39           ;Init micro words to IPR
                        0159  2516         LOOP    I,1,16                      ;Init LOOP count
                        0160  2517         LDFIELD 5$,I,Byte,3$,34,37          ;Select R no. for data
                        0171  2518         LDFIELD 5$,I,Byte,4$,34,37          ;Select R no. for data
                        0182  2519         LOOP    J,1,3                       ;Init LOOP count
                        0189  2520         SA01PAT J,2$,Longlit,6$             ;Select data pattern
                        0191  2521         LOADDCS 1$,,0,6                     ;Load test micro words
                        0198  2522         ERRLOOP                             ;Set error loop
                        019A  2523         FETCH   0                           ;Get first micro instruction
                        019F  2524         BRSTCLK 5                           ;Clock stop location 5
                        01A3  2525         CMPREG  WHREG,6$,,Long              ;Check the results
                        01AC  2526         IFERROR ,,<<DPM>>                   ;Continue if no error else call out DPM
                        01B2  2527         ENDLOOP J                           ;Get next pattern
                        01B5  2528         ENDLOOP I                           ;Get next address
                        01B8  2529         SKIP                                ;End of test
                        01BF  2530
                        01BF  2531 BEGIN_TEST_DATA
                        01BF
                        01BF       ;-------------------------------------------------------------------------
                        01BF  2532 ;+
                        01BF  2533 ; The following micro words are used to write and read the scratchpads
                        01BF  2534 ;-
                        01BF  2535
                        01BF  2536 1$:
U 00, 7700,C800,0364,0300,0470,1801     01BF  2537 ;* 00    NOP                             ;NOP Micro instruction
                        01CA  2541 2$:
U 01, 7700,7800,7FFF,FFFF,8470,1802     01CA  2542 ;* 01    LONLIT_[0]                      ;Load data pattern
U 02, 7700,4800,03C7,03D4,0470,1803     01D5  2546 ;* 02    Q_LONLIT                        ;Load Q register
```

```
                                01E0  2550 3$:
U 03, 7700,C840,02C7,0300,0470,1804  01E0  2551 ;% 03   R[0]_Q Q_D              ;Load selected register
                                01EB  2555 4$:
U 04, 6700,4800,5BE4,0300,0470,1805  01EB  2556 ;% 04   RDM_R[0]               ;Read selected register
U 05, 7300,4800,0364,0300,0470,1806  01F6  2560 ;% 05   NOP,STOP.CLOCK         ;NOP Micro instruction
                                     0201  2564
                                00   0201  2565 5$:     .Byte   0             ;Selects 0
                                01   0202  2566         .Byte   1             ;Selects 1
                                02   0203  2567         .Byte   2             ;Selects 2
                                03   0204  2568         .Byte   3             ;Selects 3
                                04   0205  2569         .Byte   4             ;Selects 4
                                05   0206  2570         .Byte   5             ;Selects 5
                                06   0207  2571         .Byte   6             ;Selects 6
                                07   0208  2572         .Byte   7             ;Selects 7
                                08   0209  2573         .Byte   8             ;Selects 8
                                09   020A  2574         .Byte   9             ;Selects 9
                                0A   020B  2575         .Byte   ^xA           ;Selects A (10)
                                0B   020C  2576         .Byte   ^xB           ;Selects B (11)
                                0C   020D  2577         .Byte   ^xC           ;Selects C (12)
                                0D   020E  2578         .Byte   ^xD           ;Selects D (13)
                                0E   020F  2579         .Byte   ^xE           ;Selects E (14)
                                0F   0210  2580         .Byte   ^xF           ;Selects F (15)
                                     0211  2581
                                     0215  2583
                                00   0215  2584 7$:     .Byte   0
                                     0216  2585
                                01   0216  2586 8$:     .Byte   1
                                     0217  2587
                                02   0217  2588 9$:     .Byte   2
                                     0218  2589
                                     0218  2590 END_TEST_DATA
                                     0218
                                     0218       ;-----------------------------------------------------------------------
                                     0218  2591
                                     0218  2592 ENDTEST
```

```
0034        .SBTTL  "TEST   013     RTEMP,GPR,IPR Scratchpad Address Integrity Test
0034  2594 ;*******************************************************************************
0034  2595 ;++
0034  2596 ;
0034  2597 ; FUNCTIONAL DESCRIPTION
0034  2598 ;
0034  2599 ;       This test consists of three subtests which check that the R Scratchpad
0034  2600 ;       address lines are not open nor shorted together.Each subtest initial-
0034  2601 ;       izes the two most significant bits of the RSRC field to one of three
0034  2602 ;       values:0,1 or 2 (0=RTEMP,1=GPR,2=IPR).Other than this initialization
0034  2603 ;       the tests are identical.
0034  2604 ;
0034  2605 ; TEST ALGORITHM
0034  2606 ;
0034  2607 ;       1. Load the micro words to perform the test.
0034  2608 ;       2. Setup loop of 20 iterations.
0034  2609 ;       3. Load RSRC Field Least Significant Byte from Table 1 into micro word
0034  2610 ;               This corresponds to the first register.
0034  2611 ;       4. Load RSRC Field Least Significant Byte from Table 2 into micro word
0034  2612 ;               This corresponds to the second register.
0034  2613 ;       5. Execute the micro sequence.
0034  2614 ;               1. Load LONLIT register with -1.
0034  2615 ;               2. Transfer data to Q register.
0034  2616 ;               3. Transfer data to first R scratchpad.
0034  2617 ;               4. Load LONLIT register with zero.
0034  2618 ;               5. Transfer data to Q register.
0034  2619 ;               6. Transfer data to second R scratchpad.
0034  2620 ;               7. Read first R scratchpad into RDM.
0034  2621 ;       6. Check the results.
0034  2622 ;
0034  2623 ; DATA TABLES
0034  2624 ;
0034  2625 ;       SUBTEST 1       SUBTEST 2       SUBTEST 3
0034  2626 ;
0034  2627 ;       Reg.  Data      Reg.  Data      Reg. Data
0034  2628 ;        .     .         .     .         .    .
0034  2629 ;        .     .         .     .         .    .
0034  2630 ;       R00<- -1        R10<- -1        R20<- -1
0034  2631 ;       R01<-  0        R11<-  0        R21<-  0
0034  2632 ;       Check R00       Check R10       Check R20
0034  2633 ;
0034  2634 ;       R00<- -1        R10<- -1        R20<- -1
0034  2635 ;       R02<-  0        R12<-  0        R22<-  0
0034  2636 ;       Check R00       Check R10       Check R20
0034  2637 ;
0034  2638 ;       R00<- -1        R10<- -1        R20<- -1
0034  2639 ;       R04<-  0        R14<-  0        R24<-  0
0034  2640 ;       Check R00       Check R10       Check R20
0034  2641 ;
0034  2642 ;       R00<- -1        R10<- -1        R20<- -1
0034  2643 ;       R08<-  0        R18<-  0        R28<-  0
0034  2644 ;       Check R00       Check R10       Check R20
0034  2645 ;
0034  2646 ;       R01<- -1        R11<- -1        R21<- -1
0034  2647 ;       R02<-  0        R12<-  0           .
```

```
                        0034  2648 ;       Check R01       Check R11          .
                        0034  2649 ;
                        0034  2650 ;          .               .               .
                        0034  2651 ;          .               .               .
                        0034  2652 ;
                        0034  2653 ;       R08<- -1        R18<- -1           .
                        0034  2654 ;       R00<-  0           .               .
                        0034  2655 ;       Check R08          .               .
                        0034  2656 ;
                        0034  2657 ;
                        0034  2658 ; DATA PATTERNS
                        0034  2659 ;
                        0034  2660 ;       FFFFFFFF
                        0034  2661 ;       00000000
                        0034  2662 ;
                        0034  2663 ; LOGIC DESCRIPTION
                        0034  2664 ;
                        0034  2665 ;       This test checks the R scratchpad RAM chips,the ALK Gate Array's
                        0034  2666 ;       address and chip select outputs and the attendant ALP Gate Array's
                        0034  2667 ;       ability to decode these instructions.
                        0034  2668 ;
                        0034  2669 ; ASSUMPTIONS
                        0034  2670 ;
                        0034  2671 ;       This test requires no signals from other CPU modules but
                        0034  2672 ;       assumes an operational RBUS and WBUS.
                        0034  2673 ;
                        0034  2674 ; ERROR DESCRIPTION
                        0034  2675 ;
                        0034  2676 ;       Data expected:FFFFFFFF
                        0034  2677 ;       Data received:
                        0034  2678 ;       Loop Count:
                        0034  2679 ;
                        0034  2680 ;--
                        0034  2681 ;*******************************************************************************
                        0034  2682
                        0034  2683         SUBTEST 1                               ;RTEMP SCRATCHPADS
                        0037
                        0037       ;////////////////////////////////////////////////////////////////////////////
                        0037  2684 ;+
                        0037  2685 ; Write -1(FFFFFFFF)into first location,zeroes in second location and
                        0037  2686 ;  read first location.
                        0037  2687 ;-
                        0037  2688
                        0037  2689         INITIALIZE                              ;DCLO Low,Init scratchpads
                        0039  2690         LDFIELD 10$,,Byte,2$,38,39              ;Select scratchpad group 00
                        004A  2691         LDFIELD 10$,,Byte,3$,38,39              ;Select scratchpad group 00
                        005B  2692         LDFIELD 10$,,Byte,4$,38,39              ;Select scratchpad group 00
                        006C  2693         LOOP    I,1,20                          ;Init loop count
                        0073  2694         LDFIELD 5$,I,Byte,2$,34,37              ;Select single scratchpad
                        0084  2695         LDFIELD 5$,I,Byte,4$,34,37              ;Select single scratchpad
                        0095  2696         LDFIELD 6$,I,Byte,3$,34,37              ;Select single scratchpad
                        00A6  2697         LOADDCS 1$,,0,9                         ;Load the test micro instrs.
                        00AD  2698         ERRLOOP                                 ;Set error loop
                        00AF  2699         FETCH   0                               ;Load first micro instr
                        00B4  2700         BRSTCLK 8                               ;Clock stop location 8
```

```
00B8  2701          CMPREG  WHREG,7$,,Long          ;Check received data
00C1  2702          IFERROR ,<<SPA>,<ALK>>          ;Continue if no error
00C8  2703                                          ;else call out SPA,ALK
00C8  2704          ENDLOOP I                       ;Continue with loop
00CB  2705
00CB  2706          SUBTEST 2                       ;GPR SCRATCHPADS
00CE
00CE        ;////////////////////////////////////////////////////////////////////////////////
00CE  2707  ;+
00CE  2708  ; Write -1(FFFFFFFF)into first location,zeroes in second location and
00CE  2709  ;  read first location.
00CE  2710  ;-
00CE  2711
00CE  2712          INITIALIZE                      ;DCLO Low,init scratchpads
00D0  2713          LDFIELD 11$,,Byte,2$,38,39      ;Select scratchpad group 10
00E1  2714          LDFIELD 11$,,Byte,3$,38,39      ;Select scratchpad group 10
00F2  2715          LDFIELD 11$,,Byte,4$,38,39      ;Select scratchpad group 10
0103  2716          LOOP    I,1,20                  ;Init first loop count
010A  2717          LDFIELD 5$,I,Byte,2$,34,37      ;Select single scratchpad
011B  2718          LDFIELD 5$,I,Byte,4$,34,37      ;Select single scratchpad
012C  2719          LDFIELD 6$,I,Byte,3$,34,37      ;Select single scratchpad
013D  2720          LOADDCS 1$,,0,9                 ;Load the test micro instrs.
0144  2721          ERRLOOP                         ;Set error loop
0146  2722          FETCH   0                       ;Load first micro instr
014B  2723          BRSTCLK 8                       ;Clock stop location 8
014F  2724          CMPREG  WHREG,7$,,Long          ;Check received data
0158  2725          IFERROR ,<<SPA>,<ALK>>          ;Continue if no error
015F  2726                                          ;else call out SPA,ALK
015F  2727          ENDLOOP I                       ;Continue with loop
0162  2728
0162  2729
0162  2730          SUBTEST 3                       ;IPR SCRATCHPADS
0165
0165        ;////////////////////////////////////////////////////////////////////////////////
0165  2731  ;+
0165  2732  ; Write -1(FFFFFFFF)into first location,zeroes in second location and
0165  2733  ;  read first location.
0165  2734  ;-
0165  2735          INITIALIZE                      ;DCLO Low,init scratchpads
0167  2736          LDFIELD 12$,,Byte,2$,38,39      ;Select scratchpad group 20
0178  2737          LDFIELD 12$,,Byte,3$,38,39      ;Select scratchpad group 20
0189  2738          LDFIELD 12$,,Byte,4$,38,39      ;Select scratchpad group 20
019A  2739          LOOP    I,1,20                  ;Init first loop count
01A1  2740          LDFIELD 5$,I,Byte,2$,34,37      ;Select single scratchpad
01B2  2741          LDFIELD 5$,I,Byte,4$,34,37      ;Select single scratchpad
01C3  2742          LDFIELD 6$,I,Byte,3$,34,37      ;Select single scratchpad
01D4  2743          LOADDCS 1$,,0,9                 ;Load test micro instrs
01DB  2744          ERRLOOP                         ;Set error loop
01DD  2745          FETCH   0                       ;Load first micro instr
01E2  2746          BRSTCLK 8                       ;Clock stop location 8
01E6  2747          CMPREG  WHREG,7$,,Long          ;Check received data
01EF  2748          IFERROR ,<<SPA>,<ALK>>          ;Continue if no error
01F6  2749                                          ;else call out SPA,ALK
01F6  2750          ENDLOOP I                       ;Continue with loop
01F9  2751          SKIP                            ;End of test
```

```
                                0200  2752
                                0200  2753 BEGIN_TEST_DATA
                                0200
                                0200       ;-------------------------------------------------------------------------
                                0200  2754 ;+
                                0200  2755 ; The following micro words are used to write and read the R Scratchpad
                                0200  2756 ; registers.
                                0200  2757 ;-
                                0200  2758
                                0200  2759 1$:
U 00, 7700,C800,0364,0300,0470,1801  0200  2760 ;% 00   NOP                          ;NOP Micro instruction
U 01, 7700,B800,0000,0000,0470,1802  020B  2764 ;% 01   LONLIT_[-1]                  ;Load LONLIT register
U 02, 7700,4800,03C7,03D4,0470,1803  0216  2768 ;% 02   Q_LONLIT                     ;Load Q register with FFFFFFFF
                                0221  2772 2$:
U 03, 7700,C840,02C7,0300,0470,1804  0221  2773 ;% 03   R[0]_Q Q_D                  ;RSRC/5$
U 04, 7700,F800,7FFF,FFFF,8470,1805  022C  2777 ;% 04   LONLIT_[0]                  ;Clear LONLIT register
U 05, 7700,4800,03C7,03D4,0470,1806  0237  2781 ;% 05   Q_LONLIT                     ;Load Q register with zero
                                0242  2785 3$:
U 06, 7700,C840,02C7,0300,0470,1807  0242  2786 ;% 06   R[0]_Q Q_D                  ;RSRC/6$
                                024D  2790 4$:
U 07, 6700,C800,5BE4,0300,0470,1808  024D  2791 ;% 07   RDM_R[0]                    ;Load RDM
U 08, 7300,4800,0364,0300,0470,1809  0258  2795 ;% 08   NOP,STOP.CLOCK              ;Stop clock,check results
                                0263  2799
                             00 0263  2800 5$:     .Byte 0                         ;Table 1
                             00 0264  2801         .Byte 0
                             00 0265  2802         .Byte 0
                             00 0266  2803         .Byte 0
                             01 0267  2804         .Byte 1
                             01 0268  2805         .Byte 1
                             01 0269  2806         .Byte 1
                             01 026A  2807         .Byte 1
                             02 026B  2808         .Byte 2
                             02 026C  2809         .Byte 2
                             02 026D  2810         .Byte 2
                             02 026E  2811         .Byte 2
                             04 026F  2812         .Byte 4
                             04 0270  2813         .Byte 4
                             04 0271  2814         .Byte 4
                             04 0272  2815         .Byte 4
                             08 0273  2816         .Byte 8
                             08 0274  2817         .Byte 8
                             08 0275  2818         .Byte 8
                             08 0276  2819         .Byte 8
                                0277  2820
                             01 0277  2821 6$:     .Byte 1                         ;Table 1
                             02 0278  2822         .Byte 2
                             04 0279  2823         .Byte 4
                             08 027A  2824         .Byte 8
                             00 027B  2825         .Byte 0
                             02 027C  2826         .Byte 2
                             04 027D  2827         .Byte 4
                             08 027E  2828         .Byte 8
                             00 027F  2829         .Byte 0
                             01 0280  2830         .Byte 1
                             04 0281  2831         .Byte 4
```

```
                    08  0282  2832          .Byte 8
                    00  0283  2833          .Byte 0
                    01  0284  2834          .Byte 1
                    02  0285  2835          .Byte 2
                    08  0286  2836          .Byte 8
                    00  0287  2837          .Byte 0
                    01  0288  2838          .Byte 1
                    02  0289  2839          .Byte 2
                    04  028A  2840          .Byte 4
                        028B  2841
                FFFFFFFF  028B  2842 7$:     .Long ^xFFFFFFFF                ;Expected data
                        028F  2843
                    00  028F  2844 10$:     .Byte 0                         ;R[0x] (RTEMP) Registers
                        0290  2845
                    01  0290  2846 11$:     .Byte 1                         ;R[1x] (GPR) Registers
                        0291  2847
                    02  0291  2848 12$:     .Byte 2                         ;R[2x] (IPR) Registers
                        0292  2849
                        0292  2850 END_TEST_DATA
                        0292
                        0292      ;-------------------------------------------------------------------------
                        0292  2851
                        0292  2852 ENDTEST
```

```
0029        .SBTTL  "TEST   014     DTYPE and ISIZE Decode VIA VBUS Test
0029  2854 ;*******************************************************************************
0029  2855 ;++
0029  2856 ;
0029  2857 ; FUNCTIONAL DESCRIPTION
0029  2858 ;
0029  2859 ;       These tests checks that portion of the DTYPE and ISIZE Decoding logic
0029  2860 ;       located on the DPM which does not involve the IRD section.
0029  2861 ;
0029  2862 ; TEST ALGORITHM
0029  2863 ;
0029  2864 ;       SUBTEST 1
0029  2865 ;
0029  2866 ;       1. Load the micro words to perform the test.
0029  2867 ;       2. Setup loop of three itrations.
0029  2868 ;       3. Select and load DTYPE micro field (ISTRM field defaults to zero).
0029  2869 ;       4. Execute the micro sequence.
0029  2870 ;               1. Load DTYPE Data into DPM.
0029  2871 ;               2. Read DTYPE Data VIA VBUS.
0029  2872 ;       5. Check the results of the VBUS read.
0029  2873 ;
0029  2874 ;       SUBTEST 2
0029  2875 ;
0029  2876 ;       1. Load the micro words to perform the test.
0029  2877 ;       2. Load ISTRM micro field with a one.
0029  2878 ;       3. Setup loop of three itrations.
0029  2879 ;       4. Select and load DTYPE micro field.
0029  2880 ;       5. Execute the micro sequence.
0029  2881 ;               1. Load DTYPE Data into DPM.
0029  2882 ;               2. Read ISTRM Data VIA VBUS.
0029  2883 ;       6. Check the results of the VBUS read.
0029  2884 ;
0029  2885 ; DATA TABLES
0029  2886 ;
0029  2887 ;               DTYPE
0029  2888 ;               1   0
0029  2889 ;
0029  2890 ;               0   0
0029  2891 ;               0   1
0029  2892 ;               1   0
0029  2893 ;               1   1
0029  2894 ;
0029  2895 ; LOGIC DESCRIPTION
0029  2896 ;
0029  2897 ;       This test checks the DSIZE and ISIZE Decoding logic located on the DPM
0029  2898 ;       that is not involved with IRD.
0029  2899 ;
0029  2900 ; ASSUMPTIONS
0029  2901 ;
0029  2902 ;       Subtest 1 assumes that the MSRC XB H signal from the MIC module is not
0029  2903 ;       asserted. Otherwise these tests require no signals from any other CPU
0029  2904 ;       modules but assume an operational VBUS.
0029  2905 ;
0029  2906 ; ERROR DESCRIPTION
0029  2907 ;
```

```
                        0029  2908 ;       SUBTEST 1
                        0029  2909 ;                              Loop 1      Loop 2      Loop 3
                        0029  2910 ;       VBUS data bit:      <24> <25>   <24> <25>   <24> <25>     ;DSIZE
                        0029  2911 ;       Data expected:        0    0      1    0      0    1
                        0029  2912 ;       Data received:
                        0029  2913 ;
                        0029  2914 ;       SUBTEST 2
                        0029  2915 ;                              Loop 1      Loop 2      Loop 3
                        0029  2916 ;       VBUS data bit:      <20> <21>   <20> <21>   <20> <21>     ;ISIZE
                        0029  2917 ;       Data expected:        1    1      0    1      1    0
                        0029  2918 ;       Data received:
                        0029  2919 ;
                        0029  2920 ;    EXAMPLE -
                        0029  2921 ;
                        0029  2922 ; ?ERROR: 0000 TEST: 0E SUBTEST: 02
                        0029  2923 ;
                        0029  2924 ; DATA:   00000120                    ;This means VBUS Bit<20> was a 1.
                        0029  2925 ;         00000020                    ;This means VBUS Bit<20> should be a 0.
                        0029  2926 ;         00000001                    ;Loop Count
                        0029  2927 ;--
                        0029  2928 ;*****************************************************************************
                        0029  2929
                        0029  2930          SUBTEST 1
                        002C
                        002C       ;////////////////////////////////////////////////////////////////////////////
                        002C  2931
                        002C  2932          INITIALIZE                      ;DCLO low,Init scratchpads
                        002E  2933          LOOP    I,1,3                   ;Set loop count
                        0035  2934          LDFIELD 4$,I,Byte,2$,40,41      ;Load DTYPE Field in microword
                        0046  2935          LOADDCS 1$,,0,3                 ;Load the test micro instrs.
                        004D  2936          ERRLOOP                         ;Set the error loop
                        004F  2937          FETCH 0                         ;Get the first micro word
                        0054  2938          BRSTCLK 2                       ;Clock stop location 2
                        0058  2939          CMPVBUS 3$,I                    ;Check the received data
                        005D  2940          IFERROR ,,<<DPM>,<MIC>>         ;Continue if no error else call out MIC
                        0064  2941          ENDLOOP I                       ;Continue with loop
                        0067  2942
                        0067  2943          SUBTEST 2
                        006A
                        006A       ;////////////////////////////////////////////////////////////////////////////
                        006A  2944
                        006A  2945          INITIALIZE                      ;DCLO Low,init scratchpads
                        006C  2946          LDFIELD 5$,,Byte,2$,33,33       ;Load ISTRM Field in microword
                        007D  2947          LOOP    I,1,3                   ;Set loop count
                        0084  2948          LDFIELD 4$,I,Byte,2$,40,41      ;Load DTYPE Field in microword
                        0095  2949          LOADDCS 1$,,0,3                 ;Load the test micro instrs.
                        009C  2950          ERRLOOP                         ;Set the error loop
                        009E  2951          FETCH 0                         ;Get the first micro word
                        00A3  2952          BRSTCLK 2                       ;Clock stop location 2
                        00A7  2953          CMPVBUS 6$,I                    ;Check the received data
                        00AC  2954          IFERROR ,,<<DPM>>               ;Continue if no error else call out DPM
                        00B2  2955          ENDLOOP I                       ;Continue with loop
                        00B5  2956          SKIP                            ;End of test
                        00BC  2957
                        00BC  2958 BEGIN_TEST_DATA
```

```
                                  00BC
                                  00BC        ;-----------------------------------------------------------------------
                                  00BC  2959  ;+
                                  00BC  2960  ; The following micro words are used to test some of the DTYPE and ISIZE data
                                  00BC  2961  ;  path thru the DPM via the VBUS.
                                  00BC  2962  ;-
                                  00BC  2963
                                  00BC  2964  1$:
U 00, 7700,C800,0364,0300,0470,1801  00BC  2965  ;% 00   NOP
                                  00C7  2969  2$:
U 01, 7600,C800,0364,0000,0470,1802  00C7  2970  ;% 01   DTYPE/0,STROBE.V.BUS          ;DTYPE Field gets 0,1 OR 2
U 02, 7300,4800,0364,0300,0470,1803  00D2  2974  ;% 02   NOP,STOP.CLOCK                ;Stop clock and check results
                                  00DD  2978
                              02  00DD  2979  3$:     .Byte 2
                                  00DE  2980          VBUSDATA <^x24>,0             ;DTYPE 0 H
                                  00DF  2981          VBUSDATA <^x25>,0             ;DTYPE 1 H
                                  00E0  2982
                              02  00E0  2983          .Byte 2
                                  00E1  2984          VBUSDATA <^x24>,1             ;DTYPE 0 H
                                  00E2  2985          VBUSDATA <^x25>,0             ;DTYPE 1 H
                                  00E3  2986
                              02  00E3  2987          .Byte 2
                                  00E4  2988          VBUSDATA <^x24>,0             ;DTYPE 0 H
                                  00E5  2989          VBUSDATA <^x25>,1             ;DTYPE 1 H
                                  00E6  2990
                              00  00E6  2991  4$:     .Byte 0
                              01  00E7  2992          .Byte 1
                              02  00E8  2993          .Byte 2
                                  00E9  2994
                              01  00E9  2995  5$:     .Byte 1                       ;ISTRM data
                                  00EA  2996
                              02  00EA  2997  6$:     .Byte 2
                                  00EB  2998          VBUSDATA <^x20>,0             ;ISIZE 0 L
                                  00EC  2999          VBUSDATA <^x21>,1             ;ISIZE 1 L
                                  00ED  3000
                              02  00ED  3001          .Byte 2
                                  00EE  3002          VBUSDATA <^x20>,1             ;ISIZE 0 L
                                  00EF  3003          VBUSDATA <^x21>,0             ;ISIZE 1 L
                                  00F0  3004
                              02  00F0  3005          .Byte 2
                                  00F1  3006          VBUSDATA <^x20>,0             ;ISIZE 0 L
                                  00F2  3007          VBUSDATA <^x21>,0             ;ISIZE 1 L
                                  00F3  3008
                                  00F3  3009  END_TEST_DATA
                                  00F3
                                  00F3        ;-----------------------------------------------------------------------
                                  00F3  3010
                                  00F3  3011  ENDTEST
```

```
003F        .SBTTL  "TEST   015     RTEMP Scratchpad Byte Write,Word Write,Longword Write Test
003F  3013 ;*******************************************************************************
003F  3014 ;++
003F  3015 ;
003F  3016 ; FUNCTIONAL DESCRIPTION
003F  3017 ;
003F  3018 ;       This test checks the scratchpads for Byte,Word and Longword
003F  3019 ;       write.
003F  3020 ;
003F  3021 ; TEST ALGORITHM
003F  3022 ;
003F  3023 ;       1. Load the micro words to perform the test.
003F  3024 ;       2. Setup loop of 3 iterations.
003F  3025 ;       3. Select and load Data type into micro word.
003F  3026 ;       4. Execute the micro sequence.
003F  3027 ;               1. Initialize R[0] to -1.
003F  3028 ;               2. Load zero pattern into R[0],data-type dependent.
003F  3029 ;               3. Read R[0] into RDM.
003F  3030 ;       5. Check the results.
003F  3031 ;       6. Looped 3 times? If not go to step three.
003F  3032 ;
003F  3033 ; DATA PATTERNS
003F  3034 ;
003F  3035 ;       FFFFFFFF
003F  3036 ;       00000000
003F  3037 ;
003F  3038 ; MICROWORD FIELDS
003F  3039 ;
003F  3040 ;       One of the following microword fields is used each time thru
003F  3041 ;       the loop:
003F  3042 ;
003F  3043 ;               SPW     DSIZE
003F  3044 ;               01       00     / Write Byte
003F  3045 ;               01       01     / Write Word
003F  3046 ;               01       10     / Write Long
003F  3047 ;
003F  3048 ;
003F  3049 ; LOGIC DESCRIPTION
003F  3050 ;
003F  3051 ;       This test checks that the ALK Gate Array and ALP Gate Arrays
003F  3052 ;       can decode the write Byte,write Word and write Longword
003F  3053 ;       instruction from the Dsize field in the micro word.
003F  3054 ;
003F  3055 ; ASSUMPTIONS
003F  3056 ;
003F  3057 ;       This test requires no signals from any other CPU modules but
003F  3058 ;       does assume an operational RBUS and WBUS.
003F  3059 ;
003F  3060 ; ERROR DESCRIPTION
003F  3061 ;
003F  3062 ;       Data expected:FFFFFF00          ;Loop count=1
003F  3063 ;                    :FFFF0000         ;Loop count=2
003F  3064 ;                    :00000000         ;Loop count=3
003F  3065 ;       Data received:
003F  3066 ;       Loop Count:
```

```
                        003F  3067 ;
                        003F  3068 ;--
                        003F  3069 ;******************************************************************************
                        003F  3070 ;//////////////////////////////////////////////////////////////////////////////
                        003F  3071 ;+
                        003F  3072 ;Load FFFFFFFF into R[0] and write 00,0000 and 00000000.
                        003F  3073 ;-
                        003F  3074         INITIALIZE                          ;DCLO Low,Init scratchpads
                        0041  3075         LOOP    I,1,3                       ;Set loop count
                        0048  3076         LDFIELD 3$,I,Byte,2$,40,41          ;Load Dtype Field
                        0059  3077         LOADDCS 1$,,0,9                     ;Load test micro instrs
                        0060  3078         ERRLOOP                             ;Set error loop
                        0062  3079         FETCH   0                           ;Fetch first micro instr
                        0067  3080         BRSTCLK 8                           ;Clock stop location 8
                        006B  3081         CMPREG  WHREG,4$,I,Long             ;Check received data
                        0074  3082         IFERROR ,<<ALP>,<ALK>>              ;Continue if no error
                        007B  3083                                             ;else call out ALP,ALK
                        007B  3084         ENDLOOP I                           ;Continue with loop
                        007E  3085         SKIP                                ;End of test
                        0085  3086
                        0085  3087 BEGIN_TEST_DATA
                        0085
                        0085       ;-----------------------------------------------------------------------
                        0085  3088 ;+
                        0085  3089 ;The following micro words load R[0] and write Dtype dependent data.
                        0085  3090 ;-
                        0085  3091
                        0085  3092 1$:
U 00, 7700,C800,0364,0300,0470,1801  0085  3093 ;% 00   NOP                     ;NOP Micro instruction
U 01, 7700,B800,0000,0000,0470,1802  0090  3097 ;% 01   LONLIT_[-1]            ;Load LONLIT Register with FFFFFFFF
U 02, 7700,4800,03C7,03D4,0470,1803  009B  3101 ;% 02   Q_LONLIT
U 03, 7700,C840,02C7,0300,0470,1804  00A6  3105 ;% 03   R[0]_Q Q_D             ;Initialize R[0]
U 04, 7700,F800,7FFF,FFFF,8470,1805  00B1  3109 ;% 04   LONLIT_[0]            ;Clear LONLIT Register
U 05, 7700,4800,03C7,03D4,0470,1806  00BC  3113 ;% 05   Q_LONLIT
                                     00C7  3117 2$:
U 06, 7701,8820,02D7,0300,0470,1807  00C7  3118 ;% 06   R[0].DT_Q              ;DTYPE/Byte,Word,Long
U 07, 6700,C800,5BE4,0300,0470,1808  00D2  3122 ;% 07   RDM_R[0]
U 08, 7300,4800,0364,0300,0470,1809  00DD  3126 ;% 08   NOP,STOP.CLOCK         ;Stop clock,check results
                                     00E8  3130
                              00     00E8  3131 3$:     .Byte   0               ;Dtype/Byte
                              01     00E9  3132         .Byte   1               ;Dtype/Word
                              02     00EA  3133         .Byte   2               ;Dtype/Long
                                     00EB  3134
                        FFFFFF00     00EB  3135 4$:     .Long   ^xFFFFFF00      ;Expected data/first read
                        FFFF0000     00EF  3136         .Long   ^xFFFF0000      ;Expected data/second read
                        00000000     00F3  3137         .Long   0               ;Expected data/third read
                                     00F7  3138
                                     00F7  3139 END_TEST_DATA
                                     00F7
                                     00F7       ;-----------------------------------------------------------------------
                                     00F7  3140
                                     00F7  3141 ENDTEST
```

```
                        0028        .SBTTL  "TEST   016     M Scratchpad Basic Operational Test
                        0028  3143 ;*****************************************************************************
                        0028  3144 ;++
                        0028  3145 ;
                        0028  3146 ; FUNCTIONAL DESCRIPTION
                        0028  3147 ;
                        0028  3148 ;       This test checks that the M scratchpad can be written and read
                        0028  3149 ;
                        0028  3150 ; TEST ALGORITHM
                        0028  3151 ;
                        0028  3152 ;       1. Load the micro words to perform the test.
                        0028  3153 ;       2. Execute the micro sequence.
                        0028  3154 ;               1. Write AAAAAAAA into MTMP0.
                        0028  3155 ;               2. Write Q register from MTMP0.
                        0028  3156 ;               3. Read Q register into RDM.
                        0028  3157 ;       3. Check the results.
                        0028  3158 ;
                        0028  3159 ; LOGIC DESCRIPTION
                        0028  3160 ;
                        0028  3161 ;       This test checks the M scratchpad write from the ALK Gate
                        0028  3162 ;       Array,the chip select from the SPA Gate Array and the MBUS
                        0028  3163 ;       leg thru the ALP Gate Arrays.
                        0028  3164 ;
                        0028  3165 ; ASSUMPTIONS
                        0028  3166 ;
                        0028  3167 ;       This test requires no signals from any other CPU modules but
                        0028  3168 ;       assumes an operational MBUS and WBUS.
                        0028  3169 ;
                        0028  3170 ; ERROR DESCRIPTION
                        0028  3171 ;
                        0028  3172 ;       Data expected:AAAAAAAA
                        0028  3173 ;       Data received:
                        0028  3174 ;
                        0028  3175 ;--
                        0028  3176 ;*****************************************************************************
                        0028  3177 ;/////////////////////////////////////////////////////////////////////////////
                        0028  3178
                        0028  3179         INITIALIZE                              ;DCLO Low,Init scratchpads
                        002A  3180         LOADREG WDREG,2$,,Long                  ;Load WBUS interface
                        0032  3181         LOADDCS 1$,,0,5                         ;Load micro instrs
                        0039  3182         ERRLOOP                                 ;Set error loop
                        003B  3183         FETCH   0                               ;Load first instruction
                        0040  3184         BRSTCLK 4                               ;Stop clock location 4
                        0044  3185         CMPREG  WHREG,2$,,Long                  ;Check received data
                        004D  3186         IFERROR ,<<SPA>,<ALK>,<ALP>>            ;Continue if no error
                        0055  3187                                                 ;else call out SPA,ALK,ALP
                        0055  3188         SKIP                                    ;End of test
                        005C  3189
                        005C  3190 BEGIN_TEST_DATA
                        005C
                        005C        ;-------------------------------------------------------------------
                        005C  3191 ;+
                        005C  3192 ; Tha following micro words are used to test the MTEMP scratchpads.
                        005C  3193 ;-
                        005C  3194
```

```
                                005C  3195 1$:
U 00, 7700,C800,0364,0300,0470,1801  005C  3196 ;% 00   NOP                          ;NOP Micro instruction
U 01, 5700,C860,0364,0300,0470,1802  0067  3200 ;% 01   M[0]_RDM
U 02, 7700,4800,5925,0300,0470,1803  0072  3204 ;% 02   Q_M[0]
U 03, 6700,8800,02C7,0300,0470,1804  007D  3208 ;% 03   RDM_Q Q_D
U 04, 7300,4800,0364,0300,0470,1805  0088  3212 ;% 04   NOP,STOP.CLOCK              ;NOP,Check results
                                0093  3216
                         AAAAAAAA  0093  3217 2$:     .Long   ^xAAAAAAAA           ;Expected data
                                0097  3218
                                0097  3219 END_TEST_DATA
                                0097
                                0097       ;-------------------------------------------------------------------------
                                0097  3220
                                0097  3221 ENDTEST
```

```
0025          .SBTTL  "TEST   017     M Scratchpad Data Integrity Test
0025  3223 ;*****************************************************************************
0025  3224 ;++
0025  3225 ;
0025  3226 ; FUNCTIONAL DESCRIPTION
0025  3227 ;
0025  3228 ;       This test checks the data integrity of the MTEMP registers by
0025  3229 ;       loading and reading six patterns.
0025  3230 ;
0025  3231 ; TEST ALGORITHM
0025  3232 ;
0025  3233 ;       1. Load micro words to perform the test.
0025  3234 ;       2. Setup a loop of six iterations.
0025  3235 ;       3. Select and load data pattern into LONLIT field of micro word.
0025  3236 ;       4. Execute the micro sequence.
0025  3237 ;               1. Load LONLIT Register with pattern.
0025  3238 ;               2. Load pattern into M[0] from LONLIT Register.
0025  3239 ;               3. Read pattern from M[0] into Q Register.
0025  3240 ;               4. Read pattern into RDM from Q register.
0025  3241 ;       5. Check the results.
0025  3242 ;       6. Looped six times? If not,go to step three.
0025  3243 ;
0025  3244 ; DATA PATTERNS
0025  3245 ;
0025  3246 ;       One of the following patterns is used each time through the loop:
0025  3247 ;
0025  3248 ;               AAAAAAAA
0025  3249 ;               55555555
0025  3250 ;               33333333
0025  3251 ;               0F0F0F0F
0025  3252 ;               00FF00FF
0025  3253 ;               0000FFFF
0025  3254 ;
0025  3255 ; LOGIC DESCRIPTION
0025  3256 ;
0025  3257 ;       This test checks that the MTEMP scratchpads' data lines are not SA0,
0025  3258 ;       SA1 nor shorted together and the AMUX to Q Register data path.
0025  3259 ;
0025  3260 ; ASSUMPTIONS
0025  3261 ;
0025  3262 ;       This test requires no signals from any other CPU modules but
0025  3263 ;       assumes an operational WBUS,MBUS and RBUS.
0025  3264 ;
0025  3265 ; ERROR DESCRIPTION
0025  3266 ;
0025  3267 ;       Data expected:Same as last written
0025  3268 ;       Data received:
0025  3269 ;       Loop Count:
0025  3270 ;
0025  3271 ;--
0025  3272 ;*****************************************************************************
0025  3273 ;/////////////////////////////////////////////////////////////////////////////
0025  3274
0025  3275          INITIALIZE                      ;DCLO Low,Init scratchpads
0027  3276          LOOP    I,1,6                   ;Initialize loop count
```

```
                        002E  3277          SA01PAT I,2$,Longlit,3$          ;Select a test pattern
                        0036  3278          LOADDCS 1$,,0,6                  ;Load the test micro instrs
                        003D  3279          ERRLOOP                          ;Set the error loop
                        003F  3280          FETCH   0                        ;Fetch the first instruction
                        0044  3281          BRSTCLK 5                        ;Clock should stop location 5
                        0048  3282          CMPREG  WHREG,3$,,Long           ;Check the received data
                        0051  3283          IFERROR ,,<<DPM>>                ;Continue if no error else call out DPM
                        0057  3284          ENDLOOP I                        ;Continue with loop
                        005A  3285          SKIP                             ;End of test
                        0061  3286
                        0061  3287 BEGIN_TEST_DATA
                        0061
                        0061              ;-------------------------------------------------------------------------
                        0061  3288 ;+
                        0061  3289 ; The following micro words are used to write and read the
                        0061  3290 ; M[0] scratchpad.
                        0061  3291 ;-
                        0061  3292
                        0061  3293 1$:
U 00, 7700,C800,0364,0300,0470,1801     0061  3294 ;% 00    NOP                              ;NOP Micro instruction
                        006C  3298 2$:
U 01, 7700,7800,7FFF,FFFF,8470,1802     006C  3299 ;% 01    LONLIT_[0]                       ;Selected pattern to LONLIT
U 02, 7700,4860,0247,03D4,0470,1803     0077  3303 ;% 02    M[0]_LONLIT                      ;Load M[0] with test pattern
U 03, 7700,C800,5925,0300,0470,1804     0082  3307 ;% 03    Q_M[0]                           ;Load Q Register
U 04, 6700,0800,02C7,0300,0470,1805     008D  3311 ;% 04    RDM_Q Q_D                        ;Load results into RDM
U 05, 7300,4800,0364,0300,0470,1806     0098  3315 ;% 05    NOP,STOP.CLOCK                   ;Stop clock and check results
                        00A3  3319
                00000000  00A3  3320 3$:      .Long                            ;Test pattern goes here
                        00A7  3321
                        00A7  3322 END_TEST_DATA
                        00A7
                        00A7              ;-------------------------------------------------------------------------
                        00A7  3323
                        00A7  3324 ENDTEST
```

```
                        0028        .SBTTL  "TEST   018     M Scratchpad Address Integrity Test
                        0028  3326 ;*******************************************************************************
                        0028  3327 ;++
                        0028  3328 ;
                        0028  3329 ; FUNCTIONAL DESCRIPTION
                        0028  3330 ;
                        0028  3331 ;       This test checks that the M Scratchpad address lines are not open nor
                        0028  3332 ;       shorted together.
                        0028  3333 ;
                        0028  3334 ; TEST ALGORITHM
                        0028  3335 ;
                        0028  3336 ;       1. Load the micro words to perform the test.
                        0028  3337 ;       2. Setup loop of 20 iterations.
                        0028  3338 ;       3. Execute the micro sequence.
                        0028  3339 ;               1. Load LONLIT register with -1.
                        0028  3340 ;               2. Transfer data to first M Scratchpad.
                        0028  3341 ;               3. Load LONLIT register with zero.
                        0028  3342 ;               4. Transfer data to second M Scratchpad.
                        0028  3343 ;               5. Read first M Scratchpad into Q Register.
                        0028  3344 ;               6. Read Q Register into RDM.
                        0028  3345 ;       4. Check the results.
                        0028  3346 ;
                        0028  3347 ; DATA TABLES
                        0028  3348 ;
                        0028  3349 ;       Reg.  Data
                        0028  3350 ;         .     .
                        0028  3351 ;       M0<-  -1
                        0028  3352 ;       M1<-   0
                        0028  3353 ;       Check M0
                        0028  3354 ;
                        0028  3355 ;       M0<-  -1
                        0028  3356 ;       M2<-   0
                        0028  3357 ;       Check M0
                        0028  3358 ;
                        0028  3359 ;       M0<-  -1
                        0028  3360 ;       M4<-   0
                        0028  3361 ;       Check M0
                        0028  3362 ;
                        0028  3363 ;       M0<-  -1
                        0028  3364 ;       M8<-   0
                        0028  3365 ;       Check M0
                        0028  3366 ;
                        0028  3367 ;       M1<-  -1
                        0028  3368 ;       M2<-   0
                        0028  3369 ;       Check M0
                        0028  3370 ;
                        0028  3371 ;           .
                        0028  3372 ;           .
                        0028  3373 ;
                        0028  3374 ;       M8<-  -1
                        0028  3375 ;       M0<-   0
                        0028  3376 ;       Check M8
                        0028  3377 ;
                        0028  3378 ;
                        0028  3379 ; DATA PATTERNS
```

```
                        0028  3380 ;
                        0028  3381 ;       FFFFFFFF
                        0028  3382 ;       00000000
                        0028  3383 ;
                        0028  3384 ; LOGIC DESCRIPTION
                        0028  3385 ;
                        0028  3386 ;       This test checks the M Scratchpad RAM chips,the ALK Gate Array's
                        0028  3387 ;       address and chip select outputs and the attendant ALP Gate Array's
                        0028  3388 ;       ability to decode these instructions.
                        0028  3389 ;
                        0028  3390 ; ASSUMPTIONS
                        0028  3391 ;
                        0028  3392 ;       This test requires no signals from other CPU modules but
                        0028  3393 ;       assumes an operational MBUS,RBUS and WBUS.
                        0028  3394 ;
                        0028  3395 ; ERROR DESCRIPTION
                        0028  3396 ;
                        0028  3397 ;       Data expected:FFFFFFFF
                        0028  3398 ;       Data received:
                        0028  3399 ;       Loop Count:
                        0028  3400 ;
                        0028  3401 ;--
                        0028  3402 ;*******************************************************************************
                        0028  3403 ;///////////////////////////////////////////////////////////////////////////////
                        0028  3404 ;+
                        0028  3405 ; Write -1(FFFFFFFF)into first location,zeroes in second location and
                        0028  3406 ;  read first location.
                        0028  3407 ;-
                        0028  3408         INITIALIZE                      ;DCLO Low,Init scratchpads
                        002A  3409         LOOP    I,1,20                  ;Init loop count
                        0031  3410         LDFIELD 5$,I,Byte,2$,64,68      ;Select single scratchpad
                        0042  3411         LDFIELD 5$,I,Byte,4$,64,68      ;Select single scratchpad
                        0053  3412         LDFIELD 6$,I,Byte,3$,64,68      ;Select single scratchpad
                        0064  3413         LOADDCS 1$,,0,8                 ;Load the test micro instrs.
                        006B  3414         ERRLOOP                         ;Set error loop
                        006D  3415         FETCH   0                       ;Load first micro instr
                        0072  3416         BRSTCLK 7                       ;Clock stop location 7
                        0076  3417         CMPREG  WHREG,7$,,Long          ;Check received data
                        007F  3418         IFERROR ,<<SPA>,<ALK>>          ;Continue if no error
                        0086  3419                                         ;else call out SPA,ALK
                        0086  3420         ENDLOOP I                       ;Continue with loop
                        0089  3421         SKIP                            ;End of test
                        0090  3422
                        0090  3423 BEGIN_TEST_DATA
                        0090
                        0090        ;-----------------------------------------------------------------------
                        0090  3424 ;+
                        0090  3425 ; The following micro words are used to write and read the M Scratchpad
                        0090  3426 ; registers.
                        0090  3427 ;-
                        0090  3428
                        0090  3429 1$:
U 00, 7700,C800,0364,0300,0470,1801     0090  3430 ;* 00    NOP                         ;NOP Micro instruction
U 01, 7700,B800,0000,0000,0470,1802     009B  3434 ;* 01    LONLIT_[-1]                 ;Load LONLIT register
                        00A6  3438 2$:
```

```
U 02. 7700,4860,0247,03D4,0470,1803  00A6  3439 ;% 02   M[0]_LONLIT                    ;ALPCTL/WX_R.Q_D,MSRC/5$
U 03. 7700,7800,7FFF,FFFF,8470,1804  00B1  3443 ;% 03   LONLIT_[0]                    ;Clear LONLIT register
                                     00BC  3447 3$:
U 04. 7700,0861,0247,03D4,0470,1805  00BC  3448 ;% 04   M[1]_LONLIT                    ;ALPCTL/WX_R.Q_D,MSRC/6$
                                     00C7  3452 4$:
U 05. 7700,4800,5925,0300,0470,1806  00C7  3453 ;% 05   Q_M[0]
U 06. 6700,8800,02C7,0300,0470,1807  00D2  3457 ;% 06   RDM_Q Q_D                     ;Load first register into RDM
U 07. 7300,C800,0364,0300,0470,1808  00DD  3461 ;% 07   NOP,STOP.CLOCK               ;Stop clock,check results
                                     00E8  3465
                                00   00E8  3466 5$:     .Byte 0                     ;Table 1
                                00   00E9  3467         .Byte 0
                                00   00EA  3468         .Byte 0
                                00   00EB  3469         .Byte 0
                                01   00EC  3470         .Byte 1
                                01   00ED  3471         .Byte 1
                                01   00EE  3472         .Byte 1
                                01   00EF  3473         .Byte 1
                                02   00F0  3474         .Byte 2
                                02   00F1  3475         .Byte 2
                                02   00F2  3476         .Byte 2
                                02   00F3  3477         .Byte 2
                                04   00F4  3478         .Byte 4
                                04   00F5  3479         .Byte 4
                                04   00F6  3480         .Byte 4
                                04   00F7  3481         .Byte 4
                                08   00F8  3482         .Byte 8
                                08   00F9  3483         .Byte 8
                                08   00FA  3484         .Byte 8
                                08   00FB  3485         .Byte 8
                                     00FC  3486
                                01   00FC  3487 6$:     .Byte 1                     ;Table 2
                                02   00FD  3488         .Byte 2
                                04   00FE  3489         .Byte 4
                                08   00FF  3490         .Byte 8
                                00   0100  3491         .Byte 0
                                02   0101  3492         .Byte 2
                                04   0102  3493         .Byte 4
                                08   0103  3494         .Byte 8
                                00   0104  3495         .Byte 0
                                01   0105  3496         .Byte 1
                                04   0106  3497         .Byte 4
                                08   0107  3498         .Byte 8
                                00   0108  3499         .Byte 0
                                01   0109  3500         .Byte 1
                                02   010A  3501         .Byte 2
                                08   010B  3502         .Byte 8
                                00   010C  3503         .Byte 0
                                01   010D  3504         .Byte 1
                                02   010E  3505         .Byte 2
                                04   010F  3506         .Byte 4
                                     0110  3507
                           FFFFFFFF  0110  3508 7$:     .Long ^xFFFFFFFF            ;Expected data
                                     0114  3509
                                     0114  3510 END_TEST_DATA
                                     0114
```

```
0114        ;-----------------------------------------------------------------------------
0114  3511
0114  3512 ENDTEST
```

```
                        0024        .SBTTL  "TEST   019     MTEMP to RTEMP Cross Write Test
                        0024  3514 ;*******************************************************************************
                        0024  3515 ;++
                        0024  3516 ;
                        0024  3517 ; FUNCTIONAL DESCRIPTION
                        0024  3518 ;
                        0024  3519 ;       This test checks that the R Scratchpad is written when the
                        0024  3520 ;       M Scratchpad is written.
                        0024  3521 ;
                        0024  3522 ; TEST ALGORITHM
                        0024  3523 ;
                        0024  3524 ;       1. Load the micro words to perform the test.
                        0024  3525 ;       2. Execute the micro sequence.
                        0024  3526 ;               1. Init R[0] with FFFFFFFF.
                        0024  3527 ;               2. Write Zero into M[0] with RSRC field set to
                        0024  3528 ;                       R[7].
                        0024  3529 ;               3. Read R[0] into RDM.
                        0024  3530 ;       3. Check the results.
                        0024  3531 ;
                        0024  3532 ; DATA PATTERN
                        0024  3533 ;
                        0024  3534 ;               FFFFFFFF
                        0024  3535 ;               00000000
                        0024  3536 ;
                        0024  3537 ; LOGIC DESCRIPTION
                        0024  3538 ;
                        0024  3539 ;       This test checks that the SPA Gate Array will write the R[0]
                        0024  3540 ;       Scratchpad simualtaneously with the M[0] Scratchpad.
                        0024  3541 ;
                        0024  3542 ; ASSUMPTIONS
                        0024  3543 ;
                        0024  3544 ;       This test requires no signals from other CPU modules but
                        0024  3545 ;       assumes an operational RBUS,WBUS and MBUS.
                        0024  3546 ;
                        0024  3547 ; ERROR DESCRIPTION
                        0024  3548 ;
                        0024  3549 ;       DATA EXPECTED:00000000
                        0024  3550 ;       DATA RECEIVED:
                        0024  3551 ;
                        0024  3552 ;--
                        0024  3553 ;*******************************************************************************
                        0024  3554 ;///////////////////////////////////////////////////////////////////////////////
                        0024  3555 ;+
                        0024  3556 ; Initialize R[0] with FFFFFFFF.Write Zero into M[0].Read R[0].
                        0024  3557 ;-
                        0024  3558         INITIALIZE                      ;DCLO Low,Init SCRATCHPAD
                        0026  3559         LOADDCS 1$,,0,<^xA>             ;Load the test micro instrs
                        002D  3560         ERRLOOP                         ;Set error loop
                        002F  3561         FETCH   0                       ;Fetch first micro instruction
                        0034  3562         BRSTCLK 9                       ;Clock stop location 9
                        0038  3563         CMPREG  WHREG,2$,,Long          ;Check received data
                        0041  3564         IFERROR ,<<SPA>>                ;Continue if no error else call out SPA
                        0047  3565         SKIP                            ;End of test
                        004E  3566
                        004E  3567 BEGIN_TEST_DATA
```

```
                                  004E
                                  004E        ;-----------------------------------------------------------------------
                                  004E  3568  ;+
                                  004E  3569  ;These micro words initialize R[0],write M[0] and read R[0].
                                  004E  3570  ;-
                                  004E  3571
                                  004E  3572  1$:
U 00, 7700,C800,0364,0300,0470,1801  004E  3573  ;% 00   NOP                          ;NOP Micro instruction
U 01, 7700,B800,0000,0000,0470,1802  0059  3577  ;% 01   LONLIT_[-1]                  ;Load LONLIT with -1
U 02, 7700,4800,03C7,03D4,0470,1803  0064  3581  ;% 02   Q_LONLIT
U 03, 7700,C840,02C7,0300,0470,1804  006F  3585  ;% 03   R[0]_Q Q_D                   ;Load M[0]
U 04, 7700,F800,7FFF,FFFF,8470,1805  007A  3589  ;% 04   LONLIT_[0]                   ;Clear LONLIT Register
U 05, 7700,4800,03C7,03D4,0470,1806  0085  3593  ;% 05   Q_LONLIT
U 06, 7700,C800,02D7,0300,0470,1807  0090  3597  ;% 06   D_Q Q_D
U 07, 7700,8860,5B24,031C,0470,1808  009B  3601  ;% 07   M[0]_D,RSRC/TEMP7
U 08, 6700,4800,5BE4,0300,0470,1809  00A6  3605  ;% 08   RDM_R[0]                     ;Load results into RDM
U 09, 7300,4800,0364,0300,0470,180A  00B1  3609  ;% 09   NOP,STOP.CLOCK               ;Stop clock,check results
                                  00BC  3613
                         00000000  00BC  3614  2$:     .Long   0                    ;Expected data
                                  00C0  3615
                                  00C0  3616  END_TEST_DATA
                                  00C0
                                  00C0        ;-----------------------------------------------------------------------
                                  00C0  3617
                                  00C0  3618  ENDTEST
```

```
                        0024        .SBTTL  "TEST   01A     RTEMP to MTEMP Cross Write Test
                        0024  3620 ;*******************************************************************************
                        0024  3621 ;++
                        0024  3622 ;
                        0024  3623 ; FUNCTIONAL DESCRIPTION
                        0024  3624 ;
                        0024  3625 ;       This test checks that the M Scratchpad is written when the R Scratchpad
                        0024  3626 ;       is written.
                        0024  3627 ;
                        0024  3628 ; TEST ALGORITHM
                        0024  3629 ;
                        0024  3630 ;       1. Load the micro words to perform the test.
                        0024  3631 ;       2. Execute the micro sequence.
                        0024  3632 ;               1. Init M[0] Scratchpad with FFFFFFFF.
                        0024  3633 ;               2. Write Zero into R[0] with MSRC field set to M[7].
                        0024  3634 ;               3. Read M[0] into RDM.
                        0024  3635 ;       3.Check the results.
                        0024  3636 ;
                        0024  3637 ; DATA PATTERN
                        0024  3638 ;
                        0024  3639 ;               FFFFFFFF
                        0024  3640 ;               00000000
                        0024  3641 ;
                        0024  3642 ; LOGIC DESCRIPTION
                        0024  3643 ;
                        0024  3644 ;       This test checks that the SPA Gate Array will write the
                        0024  3645 ;       M[0] Scratchpad simualtaneously with R[0] Scratchpad.
                        0024  3646 ;
                        0024  3647 ; ASSUMPTIONS
                        0024  3648 ;
                        0024  3649 ;       This test requires no signals from any other CPU modules but
                        0024  3650 ;       assumes an operational RBUS,WBUS and MBUS.
                        0024  3651 ;
                        0024  3652 ; ERROR DESCRIPTION
                        0024  3653 ;
                        0024  3654 ;       DATA EXPECTED:00000000
                        0024  3655 ;       DATA RECEIVED:
                        0024  3656 ;
                        0024  3657 ;--
                        0024  3658 ;*******************************************************************************
                        0024  3659 ;///////////////////////////////////////////////////////////////////////////////
                        0024  3660 ;+
                        0024  3661 ; Initialize M[0] with FFFFFFFF.Write Zero into R[0].Read M[0].
                        0024  3662 ;-
                        0024  3663         INITIALIZE                      ;DCLO Low,Init SCRATCHPADS
                        0026  3664         LOADDCS 1$,,0,<^xA>             ;Load the test micro instrs
                        002D  3665         ERRLOOP                         ;Set error loop
                        002F  3666         FETCH   0                       ;Fetch first micro instruction
                        0034  3667         BRSTCLK 9                       ;Clock stop location 9
                        0038  3668         CMPREG  WHREG,2$,,Long          ;Check received data
                        0041  3669         IFERROR ,<<SPA>>                ;Continue if no error else call out SPA
                        0047  3670         SKIP                            ;End of test
                        004E  3671
                        004E  3672 BEGIN_TEST_DATA
                        004E
```

```
                                  004E        ;-----------------------------------------------------------------------
                                  004E  3673  ;+
                                  004E  3674  ;These micro words initialize M[0],write R[0] and read M[0].
                                  004E  3675  ;-
                                  004E  3676
                                  004E  3677  1$:
U 00, 7700,C800,0364,0300,0470,1801  004E  3678  ;% 00   NOP                          ;NOP Micro instruction
U 01, 7700,B800,0000,0000,0470,1802  0059  3682  ;% 01   LONLIT_[-1]                  ;Load LONLIT with -1
U 02, 7700,4860,0247,03D4,0470,1803  0064  3686  ;% 02   M[0]_LONLIT                  ;Load M[0]
U 03, 7700,7800,7FFF,FFFF,8470,1804  006F  3690  ;% 03   LONLIT_[0]                   ;Clear LONLIT Register
U 04, 7700,4800,03C7,03D4,0470,1805  007A  3694  ;% 04   Q_LONLIT                     ;Clear Q Register
U 05, 7700,0847,02C7,0300,0470,1806  0085  3698  ;% 05   R[0]_Q Q_D,MSRC/TEMP7        ;Clear R[0],MSRC/TEMP7
U 06, 7700,B800,0000,0000,0470,1807  0090  3702  ;% 06   LONLIT_[-1]                  ;Load LONLIT with -1
U 07, 7700,C800,03C7,03D4,0470,1808  009B  3706  ;% 07   Q_LONLIT                     ;Clear 0's from Q Register
U 08, 6700,4800,5925,0300,0470,1809  00A6  3710  ;% 08   RDM_WB,Q_M[0]                ;Read M[0] into RDM
U 09, 7300,4800,0364,0300,0470,180A  00B1  3714  ;% 09   NOP,STOP.CLOCK               ;Stop clock,check results
                                  00BC  3718
                         00000000  00BC  3719  2$:     .Long   0                    ;Expected data
                                  00C0  3720
                                  00C0  3721  END_TEST_DATA
                                  00C0
                                  00C0        ;-----------------------------------------------------------------------
                                  00C0  3722
                                  00C0  3723  ENDTEST
```

```
                        002D        .SBTTL  "TEST   01B     ALP Gate Array Q Register Input Mux Test
                        002D  3725 ;*******************************************************************************
                        002D  3726 ;++
                        002D  3727 ;
                        002D  3728 ; FUNCTIONAL DESCRIPTION
                        002D  3729 ;
                        002D  3730 ;       This test checks that the input Mux's of the Q Register are operational
                        002D  3731 ;       by shifting the Q Register left and then shifting the Q Register right.
                        002D  3732 ;
                        002D  3733 ; TEST ALGORITHM
                        002D  3734 ;
                        002D  3735 ;       1. Load the micro words to perform the test.
                        002D  3736 ;       2. Setup a loop of four iterations.
                        002D  3737 ;       3. Load RSRC Field to load Q Register with test pattern.
                        002D  3738 ;       4. Load ALPCTL Field with Q Register shift.
                        002D  3739 ;               1. Execute the micro sequence.
                        002D  3740 ;               2. Load Q Register with test pattern.
                        002D  3741 ;               3. Shift Q Register with A Mux input and WMUX input set to
                        002D  3742 ;                       FFFFFFFF.
                        002D  3743 ;       5. Check the results of the Q Register shift.
                        002D  3744 ;
                        002D  3745 ; DATA PATTERNS
                        002D  3746 ;
                        002D  3747 ;               AAAAAAAA
                        002D  3748 ;               FFFFFFFF
                        002D  3749 ;               55555555
                        002D  3750 ;
                        002D  3751 ; LOGIC DESCRIPTION
                        002D  3752 ;
                        002D  3753 ;       This test checks the ALP Gate Array's Q Register input Mux's which are
                        002D  3754 ;       comprised of Nand Gates.All gates are tested.
                        002D  3755 ;
                        002D  3756 ; ASSUMPTIONS
                        002D  3757 ;
                        002D  3758 ;       This test requires no signals from any other CPU modules but assumes an
                        002D  3759 ;       operational RBUS, MBUS, WBUS, CCC Gate array and CLA Gate array.
                        002D  3760 ;
                        002D  3761 ; ERROR DESCRIPTION
                        002D  3762 ;
                        002D  3763 ;       Data expected:  Loop 1  55555555
                        002D  3764 ;                       Loop 2  AAAAAAAA
                        002D  3765 ;                       Loop 3  AAAAAAAA
                        002D  3766 ;                       Loop 4  55555555
                        002D  3767 ;       Data received:
                        002D  3768 ;       Loop Count:
                        002D  3769 ;--
                        002D  3770 ;*******************************************************************************
                        002D  3771 ;///////////////////////////////////////////////////////////////////////////////
                        002D  3772
                        002D  3773         INITIALIZE                          ;DCLO low,Init scratchpads
                        002F  3774         LOOP    I,1,4                       ;Set loop count
                        0036  3775         LDFIELD 6$,I,Byte,2$,34,39          ;Load RSRC Field to load Q Register
                        0047  3776         LDFIELD 5$,I,Word,3$,48,63          ;Load ALPCTL and ROT Field
                        0058  3777         LOADDCS 1$,,0,5                     ;Load the test micro instrs.
                        005F  3778         ERRLOOP                             ;Set the error loop
```

```
                                0061  3779          FETCH   0                       ;Get the first micro word
                                0066  3780          BRSTCLK 4                       ;Clock stop location 4
                                006A  3781          CMPREG  WHREG,4$,I,Long         ;Check the received data
                                0073  3782          IFERROR ,<<ALP>,<ALK>,<SRM>,<SRK>>      ;Continue if no error else call
                                007C  3783                                          ; out ALP,ALK,SRM,SRK
                                007C  3784          ENDLOOP I                       ;Continue with loop
                                007F  3785          SKIP                            ;End of test
                                0086  3786
                                0086  3787 BEGIN_TEST_DATA
                                0086  3788 ;+
                                0086  3789 ;The following micro words are used to test the Q Register's Input Mux's, which
                                0086  3790 ;are contained within the ALP Gate Array.
                                0086  3791 ;-
                                0086  3792
                                0086  3793 1$:
U 00, 7700,C800,0364,0300,0470,1801  0086  3794 ;% 00    NOP
                                0091  3798 2$:
U 01, 7700,8800,5BE5,0300,0470,1802  0091  3799 ;% 01    Q_R[0]
                                009C  3803 3$:
U 02, 7700,4801,0364,0308,0470,1803  009C  3804 ;% 02    ALPCTL/NOP,RSRC/TEMP2,MSRC/TEMP1 ;Test Micro Word
U 03, 6700,8800,02C7,0300,0470,1804  00A7  3808 ;% 03    RDM_Q Q_D
U 04, 7300,4800,0364,0300,0470,1805  00B2  3812 ;% 04    NOP,STOP.CLOCK                  ;Stop clock and check results
                                00BD  3816
                        55555555  00BD  3817 4$:      .Long   ^x55555555              ;Check data
                        AAAAAAAA  00C1  3818         .Long   ^xAAAAAAAA              ;Check data
                        AAAAAAAA  00C5  3819         .Long   ^xAAAAAAAA              ;Check data
                        55555555  00C9  3820         .Long   ^x55555555              ;Check data
                                00CD  3821
                            1042  00CD  3822 5$:      .Word   ^x1042                  ;ALU/A-B-CI,MUX/M.R2,DQ2/SQL,ROT/QSI=1
                            0042  00CF  3823         .Word   ^x42                    ;ALU/A-B-CI,MUX/M.R2,DQ2/SQL,ROT/QSI=0
                            1043  00D1  3824         .Word   ^x1043                  ;ALU/A-B-CI,MUX/M.R2,DQ2/SQR,ROT/QSI=1
                            0043  00D3  3825         .Word   ^x43                    ;ALU/A-B-CI,MUX/M.R2,DQ2/SQR,ROT/QSI=0
                                00D5  3826
                              04  00D5  3827 6$:      .Byte   4                       ;RSRC Field to load Q Register
                              03  00D6  3828         .Byte   3                       ;RSRC Field to load Q Register
                              03  00D7  3829         .Byte   3                       ;RSRC Field to load Q Register
                              04  00D8  3830         .Byte   4                       ;RSRC Field to load Q Register
                                00D9  3831
                                00D9  3832 BEGIN_CPU_DATA
                                0002  3833
                                0002  3834 RTEMP_DATA
                                0001  3835
                        00000000  0001  3836         .Long     0             ;TEMP0
                        FFFFFFFF  0005  3837         .Long   ^xFFFFFFFF      ;TEMP1
                        00000000  0009  3838         .Long     0             ;TEMP2
                        55555555  000D  3839         .Long   ^x55555555      ;TEMP3
                        AAAAAAAA  0011  3840         .Long   ^xAAAAAAAA      ;TEMP4
                                0015  3841
                                0015  3842 END_CPU_DATA
                                00D9  3843
                                00D9  3844 END_TEST_DATA
                                00D9
                                00D9        ;-------------------------------------------------------------------------
```

```
                        00D9  3845
                        00D9  3846 ENDTEST
```

```
                        .SBTTL  "TEST   01C     ALU Zero and Sign Extend MBUS Data Test
002C  3848 ;*******************************************************************************
002C  3849 ;++
002C  3850 ;
002C  3851 ; FUNCTIONAL DESCRIPTION
002C  3852 ;
002C  3853 ;       This test checks the ZERO and SIGN EXTEND functions of the ALU.
002C  3854 ;
002C  3855 ; TEST ALGORITHM
002C  3856 ;
002C  3857 ;       1. Load the micro words to perform the test.
002C  3858 ;       2. Setup loop of eight iterations.
002C  3859 ;       3. Select and load DTYPE, ROTATE, RSRC and MSRC micro fields.
002C  3860 ;       4. Execute the micro sequence.
002C  3861 ;               1. Load D and Q Register.
002C  3862 ;               2. Extend data onto WBUS
002C  3863 ;       5. Check the results of the WBUS read.
002C  3864 ;
002C  3865 ; DATA TABLES
002C  3866 ;
002C  3867 ;                       DTYPE           ROTATE <5>
002C  3868 ;                         0     Byte      0       Zero extend
002C  3869 ;                         1     Word      0       Zero extend
002C  3870 ;                         0     Byte      0       Zero extend
002C  3871 ;                         1     Word      0       Zero extend
002C  3872 ;                         0     Byte      1       Sign extend
002C  3873 ;                         1     Word      1       Sign extend
002C  3874 ;                         0     Byte      1       Sign extend
002C  3875 ;                         1     Word      1       Sign extend
002C  3876 ;
002C  3877 ; LOGIC DESCRIPTION
002C  3878 ;
002C  3879 ;       This test checks the AMUX Gates in the ALP Gate Array,the BYTE L signal
002C  3880 ;       from the ALK Gate Array and various logic on the DPM.
002C  3881 ;
002C  3882 ; ASSUMPTIONS
002C  3883 ;
002C  3884 ;       This test requires no signals from any other CPU modules but assumes an
002C  3885 ;       operational MBUS and RBUS.
002C  3886 ;
002C  3887 ; ERROR DESCRIPTION
002C  3888 ;
002C  3889 ;       Data expected:  LOOP 1          FF
002C  3890 ;                       LOOP 2          FFFF
002C  3891 ;                       LOOP 3          7F
002C  3892 ;                       LOOP 4          7F7F
002C  3893 ;                       LOOP 5          FFFFFF80
002C  3894 ;                       LOOP 6          FFFF8080
002C  3895 ;       Data received:
002C  3896 ;--
002C  3897 ;*******************************************************************************
002C  3898 ;///////////////////////////////////////////////////////////////////////////////
002C  3899
002C  3900          INITIALIZE                      ;DCLO low,Init scratchpads
002E  3901          LOADREG WDREG,9$,,Long          ;Load data for MTEMP8
```

```
                        0036  3902          LOOP    I,1,6                   ;Set loop count
                        003D  3903          LDFIELD 4$,I,Byte,2$,40,41      ;Load DTYPE Field in microword
                        004E  3904          LDFIELD 5$,I,Byte,2$,58,63      ;Load ROTATE Field in microword
                        005F  3905          LDFIELD 6$,I,Byte,7$,34,39      ;Load RSRC Field in microword
                        0070  3906          LDFIELD 8$,I,Byte,2$,64,68      ;Load MSRC Field in microword
                        0081  3907          LOADDCS 1$,,0,4                 ;Load the test micro instrs.
                        0088  3908          ERRLOOP                         ;Set the error loop
                        008A  3909          FETCH 0                         ;Get the first micro word
                        008F  3910          BRSTCLK 4                       ;Clock stop location 4
                        0093  3911          CMPREG  WHREG,3$,I,Long         ;Check the received data
                        009C  3912          IFERROR ,<<ALK>,<ALP>,<SRK>,<SRM>>      ;Continue if no error else
                        00A5  3913                                          ;call out ALK,ALP
                        00A5  3914          ENDLOOP I                       ;Continue with loop
                        00A8  3915          SKIP                            ;End of test
                        00AF  3916
                        00AF  3917 BEGIN_TEST_DATA
                        00AF
                        00AF        ;-------------------------------------------------------------------------
                        00AF  3918 ;+
                        00AF  3919 ; The following micro words are used to test the EXTEND MBUS Function.
                        00AF  3920 ;-
                        00AF  3921
                        00AF  3922 1$:
U 00, 7700,C800,0364,0300,0470,1801     00AF  3923 ;% 00    NOP
U 01, 5701,C868,0364,0200,0470,1802     00BA  3927 ;% 01    WB_RDM,MSRC/TEMP8,SPW/MLONG,SIZE[LONG]  ;Load MTEMP8 with FFFF7F7F
                        00C5  3931 7$:
U 02, 7700,4801,0057,0308,0470,1803     00C5  3932 ;% 02    ALPCTL/WX_D_R.Q_M,RSRC/TEMP2,MSRC/TEMP1
                        00D0  3936 2$:
U 03, 6701,8802,0190,0000,0470,1804     00D0  3937 ;% 03    ALU/A+B+CI,MUX/XM.Q,DTYPE/Byte,ROT/XZ.MR,MSRC/TEMP2,RDM_WB,VSIZE/1
U 04, 7300,4800,0364,0300,0470,1805     00DB  3941 ;% 04    NOP,STOP.CLOCK                  ; Stop clock and check results
                        00E6  3945
                        00E6  3946 ; Expected data
                        00E6  3947
                000000FF  00E6  3948 3$:      .Long   ^x FF                   ; Zero extended byte
                0000FFFF  00EA  3949          .Long   ^x FFFF                 ;    "       "    word
                0000007F  00EE  3950          .Long   ^x 7F                   ;    "       "    byte
                00007F7F  00F2  3951          .Long   ^x 7F7F                 ;    "       "    word
                0000007F  00F6  3952          .Long   ^x 7F                   ; Sign extended byte
                00007F7F  00FA  3953          .Long   ^x 7F7F                 ;    "       "    word
                FFFFFF80  00FE  3954          .Long   ^x FFFFFF80             ;    "       "    byte
                FFFF8080  0102  3955          .Long   ^x FFFF8080             ;    "       "    word
                          0106  3956
                          0106  3957 ; DTYPE data
                          0106  3958
                       00 0106  3959 4$:      .Byte   0                       ; Byte
                       01 0107  3960          .Byte   1                       ; Word
                       00 0108  3961          .Byte   0
                       01 0109  3962          .Byte   1
                       00 010A  3963          .Byte   0
                       01 010B  3964          .Byte   1
                       00 010C  3965          .Byte   0
                       01 010D  3966          .Byte   1
                          010E  3967
                          010E  3968 ; ROTATE field data
                          010E  3969
```

```
                     00  010E  3970 5$:      .Byte   0                       ; Zero extend
                     00  010F  3971          .Byte   0
                     00  0110  3972          .Byte   0
                     00  0111  3973          .Byte   0
                     20  0112  3974          .Byte   ^x 20                   ; Sign extend
                     20  0113  3975          .Byte   ^x 20
                     20  0114  3976          .Byte   ^x 20
                     20  0115  3977          .Byte   ^x 20
                         0116  3978
                     02  0116  3979 6$:      .Byte   2                       ; D Register_TEMP2
                     02  0117  3980          .Byte   2
                     01  0118  3981          .Byte   1                       ; D Register_TEMP1
                     01  0119  3982          .Byte   1
                     02  011A  3983          .Byte   2                       ; D Register_TEMP2
                     02  011B  3984          .Byte   2
                     01  011C  3985          .Byte   1                       ; D Register_TEMP1
                     01  011D  3986          .Byte   1
                         011E  3987
                         011E  3988 ; MSRC data - to be extended
                         011E  3989
                     02  011E  3990 8$:      .Byte   2                       ; TEMP2
                     02  011F  3991          .Byte   2                       ; TEMP2
                     07  0120  3992          .Byte   7                       ; TEMP7
                     08  0121  3993          .Byte   8                       ; TEMP8
                     03  0122  3994          .Byte   3                       ; TEMP3
                     04  0123  3995          .Byte   4                       ; TEMP4
                     05  0124  3996          .Byte   5                       ; TEMP5
                     06  0125  3997          .Byte   6                       ; TEMP6
                         0126  3998
                 FFFF7F7F  0126  3999 9$:      .Long   ^xFFFF7F7F              ; MTEMP8 data
                         012A  4000
                         012A  4001 BEGIN_CPU_DATA
                         0002  4002
                         0002  4003 RTEMP_DATA
                         0001  4004
                 00000000  0001  4005          .Long   0                       ; TEMP0 - IGNORE
                 00000000  0005  4006          .Long   0                       ; TEMP1
                 FFFFFFFF  0009  4007          .Long   ^xFFFFFFFF              ; TEMP2
                 0000007F  000D  4008          .Long   ^x7F                    ; TEMP3
                 00007F7F  0011  4009          .Long   ^x7F7F                  ; TEMP4
                 00000080  0015  4010          .Long   ^x80                    ; TEMP5
                 00008080  0019  4011          .Long   ^x8080                  ; TEMP6
                 FFFFFF7F  001D  4012          .Long   ^xFFFFFF7F              ; TEMP7
                         0021  4013
                         0021  4014 END_CPU_DATA
                         012A  4015
                         012A  4016 END_TEST_DATA
                         012A
                         012A       ;-----------------------------------------------------------------------------
                         012A  4017
                         012A  4018 ENDTEST
```

```
                        0017        .SBTTL  "TEST   01D     ALU Functions Test
                        0017  4020 ;*******************************************************************************
                        0017  4021 ;++
                        0017  4022 ;
                        0017  4023 ; FUNCTIONAL DESCRIPTION
                        0017  4024 ;
                        0017  4025 ;       This test consists of two subtests that check the ALU and alu function
                        0017  4026 ;       decode logic in the ALK and ALP Gate Arrays as determined by the XLOG
                        0017  4027 ;       simulator.These tests are table driven, with the 'A' operands,'B'
                        0017  4028 ;       operands,ALU Functions and the expected results all in tables listed
                        0017  4029 ;       below.
                        0017  4030 ;
                        0017  4031 ; TEST ALGORITHM
                        0017  4032 ;
                        0017  4033 ;  SUBTEST 1
                        0017  4034 ;
                        0017  4035 ;       1. Load the ROT field with zero. (No carry in)
                        0017  4036 ;       2. Setup a loop of 69 iterations.
                        0017  4037 ;       3. Put the 'A' operand into Micro Instruction 0's LONLIT Field.
                        0017  4038 ;       4. Put the 'B' operand into Micro Instruction 3's LONLIT Field.
                        0017  4039 ;       5. Put the ALU function into Micro Instruction 6.
                        0017  4040 ;       6. Load the 9 Micro Instructions into DCS.
                        0017  4041 ;       7. Execute the Micro Sequence.
                        0017  4042 ;               1. Load the LONLIT register with the 'A' operand.
                        0017  4043 ;               2. Load the Q Register from the LONLIT Register.
                        0017  4044 ;               3. Load M[0] from Q Register,M[0] now holds 'A' operand.
                        0017  4045 ;               4. Load the LONLIT Register with the 'B' operand.
                        0017  4046 ;               5. Load the Q Register from the LONLIT Register.
                        0017  4047 ;               6. Load R[1] from Q Register,R[1] now holds 'B' operand.
                        0017  4048 ;               7. Execute Q_M[0] X R[1], where X is the alu function being
                        0017  4049 ;                       tested,leave the result in the Q Register.
                        0017  4050 ;               8. Read the Q Register into the RDM.
                        0017  4051 ;       8. Check the results.
                        0017  4052 ;       9. Looped 69 times?If not, go to step 2.
                        0017  4053 ;
                        0017  4054 ;  SUBTEST 2
                        0017  4055 ;
                        0017  4056 ;       1. Load ROT field with 2. (Carry in)
                        0017  4057 ;       2. Setup a loop of 19 iterations.
                        0017  4058 ;       3. Put the 'A' operand into Micro Instruction 0's LONLIT Field.
                        0017  4059 ;       4. Put the 'B' operand into Micro Instruction 3's LONLIT Field.
                        0017  4060 ;       5. Put the ALU function into Micro Instruction 6.
                        0017  4061 ;       6. Load the 9 Micro Instructions into DCS.
                        0017  4062 ;       7. Execute the Micro Sequence.
                        0017  4063 ;               1. Load the LONLIT register with the 'A' operand.
                        0017  4064 ;               2. Load the Q Register from the LONLIT Register.
                        0017  4065 ;               3. Load M[0] from Q Register,M[0] now holds 'A' operand.
                        0017  4066 ;               4. Load the LONLIT Register with the 'B' operand.
                        0017  4067 ;               5. Load the Q Register from the LONLIT Register.
                        0017  4068 ;               6. Load R[1] from Q Register,R[1] now holds 'B' operand.
                        0017  4069 ;               7. Execute Q_M[0] X R[1], where X is the alu function being
                        0017  4070 ;                       tested,leave the result in the Q Register.
                        0017  4071 ;               8. Read the Q Register into the RDM.
                        0017  4072 ;       8. Check the results.
                        0017  4073 ;       9. Looped 19 times?If not, go to step 2.
```

```
0017  4074 ;
0017  4075 ; DATA PATTERNS
0017  4076 ;
0017  4077 ;       This table lists the cases used in this test.
0017  4078 ;
0017  4079 ;  SUBTEST 1
0017  4080 ;
0017  4081 ;          Table 1          Table 2          Table 3          Table 4
0017  4082 ;      'A' operand      'B' operand      ALU function  Expected result
0017  4083 ;        11111111         08080808         A-B-CI           09090909
0017  4084 ;        11111111         00808080         A-B-CI           10909091
0017  4085 ;        41414141         60606060         A-B-CI           E0E0E0E1
0017  4086 ;        01010101         40404040         A-B-CI           C0C0C0C1
0017  4087 ;        B1B1B1B1         C0C0C0C0         A-B-CI.BCD       91909090
0017  4088 ;        99999999         08080808         A-B-CI.BCD       91919191
0017  4089 ;        01010101         80808080         A-B-CI.BCD       21202020
0017  4090 ;        21212121         62626262         A-B-CI.BCD       59585858
0017  4091 ;        12121212         26262606         A-B-CI.BCD       86858505
0017  4092 ;        91919191         68686868         A-B-CI.BCD       23232323
0017  4093 ;        19191919         86868606         A-B-CI.BCD       33323212
0017  4094 ;        01010101         88888888         A-B-CI.BCD       13121212
0017  4095 ;        10101010         88888808         A-B-CI.BCD       22212101
0017  4096 ;        08080808         00000000         A-B-CI.SL        10101010
0017  4097 ;        41414141         40404040         A-B-CI.SR        00808080
0017  4098 ;        01010101         10101010         A-B-CI.SR        78787878
0017  4099 ;        41414141         30303030         A-B-CI.SR        08888888
0017  4100 ;        71717171         58585858         A-B-CI.SR        0C8C8C8C
0017  4101 ;        1B1B1B1B         0C0C0C0C         A-B-CI.BCD       09090909
0017  4102 ;        B0B0B0B0         61616161         B-A-CI           B0B0B0B1
0017  4103 ;        40404040         B1B1B1B1         B-A-CI           71717171
0017  4104 ;        08080808         01010101         B-A-CI           F8F8F8F9
0017  4105 ;        00808080         10101010         B-A-CI           0F8F8F90
0017  4106 ;        C8C8C8C8         C1C1C1C1         B-A-CI           F8F8F819
0017  4107 ;        18181818         71717171         B-A-CI           59595959
0017  4108 ;        01818181         17171717         B-A-CI           15959596
0017  4109 ;        A8A8A8A8         98989898         A+B+CI           41414140
0017  4110 ;        90909090         30303030         A+B+CI           C0C0C0C0
0017  4111 ;        09090909         03030303         A+B+CI           0C0C0C0C
0017  4112 ;        50505050         90909090         A+B+CI.BCD       41414140
0017  4113 ;        05050505         09090909         A+B+CI.BCD       14141414
0017  4114 ;        30303030         60606060         A+B+CI.BCD       90909090
0017  4115 ;        03030303         06060606         A+B+CI.BCD       09090909
0017  4116 ;        42424242         58585858         A+B+CI.BCD       01010100
0017  4117 ;        65656565         15151515         A+B+CI.BCD       80808080
0017  4118 ;        66666666         E4E4E4E4         A+B+CI.BCD       30313131
0017  4119 ;        84848484         06060606         A+B+CI.BCD       90909090
0017  4120 ;        10101010         50505050         A+B+CI.BCD       60606060
0017  4121 ;        01010101         05050505         A+B+CI.BCD       06060606
0017  4122 ;        30303030         30303030         A+B+CI.BCD       60606060
0017  4123 ;        03030303         03030303         A+B+CI.BCD       06060606
0017  4124 ;        23232323         37373737         A+B+CI.BCD       60606060
0017  4125 ;        80808080         80808080         A+B+CI.BCD       61616160
0017  4126 ;        08080808         08080808         A+B+CI.BCD       16161616
0017  4127 ;        30303030         70707070         A+B+CI.BCD       01010100
0017  4128 ;        03030303         07070707         A+B+CI.BCD       10101010
```

```
0017  4129 ;        87878787        23232323        A+B+CI.BCD      11111110
0017  4130 ;        50505050        D0D0D0D0        A+B+CI.BCD      81818180
0017  4131 ;        05050505        0D0D0D0D        A+B+CI.BCD      18181818
0017  4132 ;        29292929        71717171        A+B+CI.BCD      01010100
0017  4133 ;        80808080        40404040        A+B+CI.BCD      20212121
0017  4134 ;        08080808        04040404        A+B+CI.BCD      12121212
0017  4135 ;        00000000        10101010        A+B+CI.SR       08080808
0017  4136 ;        00000000        01010101        A+B+CI.SR       00808080
0017  4137 ;        C8C8C8C8        68686868        A+B+CI.SL       62626260
0017  4138 ;        18181818        D8D8D8D8        A+B+CI.SL       E1E1E1E0
0017  4139 ;        00000000        10101010        A.AND.B         00000000
0017  4140 ;        00000000        01010101        A.AND.B         00000000
0017  4141 ;        D8D8D8D8        B8B8B8B8        (A.AND.B).SL    31313130
0017  4142 ;        A0A0A0A0        D0D0D0D0        A.AND(.NOT.B)   20202020
0017  4143 ;        0A0A0A0A        0D0D0D0D        A.AND(.NOT.B)   02020202
0017  4144 ;        21212121        01010101        (A.AND.B).SR    00808080
0017  4145 ;        12121212        10101010        (A.AND.B).SR    08080808
0017  4146 ;        80808080        F0F0F0F0        (.NOT.A).AND.B  70707070
0017  4147 ;        08080808        0F0F0F0F        (.NOT.A).AND.B  07070707
0017  4148 ;        10101010        20202020        A.OR.B          30303030
0017  4149 ;        01010101        02020202        A.OR.B          03030303
0017  4150 ;        00000000        80808080        A.XOR.B         80808080
0017  4151 ;        00000000        08080808        A.XOR.B         08080808
0017  4152 ;
0017  4153 ;  SUBTEST 2
0017  4154 ;
0017  4155 ;          Table 5         Table 6         Table 6         Table 8
0017  4156 ;      'A' operand     'B' operand     ALU function  Expected result
0017  4157 ;        14141414        06060606        A-B-CI          0D0D0D0C
0017  4158 ;        10101010        04040404        A-B-CI          0C0C0C0B
0017  4159 ;        17171717        05858585        A-B-CI.SR       08C8C8C8
0017  4160 ;        19999999        00808080        A-B-CI.BCD      19191918
0017  4161 ;        10101010        08080808        A-B-CI.BCD      02020201
0017  4162 ;        10101010        01010101        A-B-CI.SR       07878783
0017  4163 ;        14141414        04040404        A-B-CI.SR       08080807
0017  4164 ;        14141414        03030303        A-B-CI.SR       08888888
0017  4165 ;        80808080        00000000        A-B-CI.SL       010100FE
0017  4166 ;        0B0B0B0B        16161616        B-A-CI          0B0B0B0A
0017  4167 ;        0C8C8C8C        1C1C1C1C        B-A-CI          0F8F8F90
0017  4168 ;        0A8A8A8A        09898989        A+B+CI          14141413
0017  4169 ;        0C8C8C8C        06868686        A+B+CI.SL       26262626
0017  4170 ;        01818181        008D8D8D        A+B+CI.SL       041E1E1E
0017  4171 ;        24242404        85858505        A+B+CI.BCD      10101010
0017  4172 ;        56565606        51515101        A+B+CI.BCD      08080808
0017  4173 ;        66666606        4E4E4E0E        A+B+CI.BCD      13131313
0017  4174 ;        48484808        60606000        A+B+CI.BCD      09090909
0017  4175 ;        78787808        32323202        A+B+CI.BCD      11111111
0017  4176 ;        92929202        17171707        A+B+CI.BCD      10101010
0017  4177 ;        0D8D8D8D        0B8B8B8B        (A.AND.B).SL    04C4C4C4
0017  4178 ;
0017  4179 ; LOGIC DESCRIPTION
0017  4180 ;
0017  4181 ;        This test checks the ALU and alu function decode logic contained
0017  4182 ;        entirely within the 8 ALP Gate Arrays and the ALK Gate Array.
0017  4183 ;
```

```
                        0017  4184 ; ASSUMPTIONS
                        0017  4185 ;
                        0017  4186 ;       This test requires no signals from other CPU modules.It assumes the
                        0017  4187 ;       previous tests have been run to verify that logic they use, such as
                        0017  4188 ;       the RBUS,MBUS,WBUS and registers,are operational.
                        0017  4189 ;
                        0017  4190 ; ERROR DESCRIPTION
                        0017  4191 ;
                        0017  4192 ;       Data Expected:          ;Subtest 1 - Table 4:Subtest 2 - Table 8
                        0017  4193 ;       Data Received:          ;Subtest 1
                        0017  4194 ;       Data Received:          ;Subtest 2
                        0017  4195 ;       Loop Count:             ;Subtest 1
                        0017  4196 ;       Loop Count:             ;Subtest 2
                        0017  4197 ;
                        0017  4198 ;--
                        0017  4199 ;*******************************************************************************
                        0017  4200
                        0017  4201          SUBTEST 1
                        001A
                        001A               ;////////////////////////////////////////////////////////////////////////////////
                        001A  4202
                        001A  4203          INITIALIZE                          ;DCLO Low,Init Scratchpads
                        001C  4204          LDFIELD 14$,,Byte,4$,58,63          ;Load ROT Field with carry-in
                        002D  4205          LOOP    I,1,69                      ;Setup loop
                        0034  4206          LDFIELD 5$,I,Long,2$,31,62,LONLIT       ;Load 'A' operand
                        0045  4207          LDFIELD 6$,I,Long,3$,31,62,LONLIT       ;Load 'B' operand
                        0056  4208          LDFIELD 7$,I,Byte,4$,50,53          ;Load test alu function
                        0067  4209          LOADDCS 1$,,0,<^xA>                 ;Load the test micro instrs.
                        006E  4210          ERRLOOP                             ;Set the error loop
                        0070  4211          FETCH   0                           ;Fetch the first micro instr.
                        0075  4212          BRSTCLK 9                           ;Clock stop location 9
                        0079  4213          CMPREG  WHREG,8$,I,Long             ;Check the received data
                        0082  4214          IFERROR ,<<ALP>,<ALK>,<CLA>,<SRM>,<SRK>>        ;Continue if no error
                        008C  4215                                              ;else call out ALP,ALK,CLA
                        008C  4216          ENDLOOP I
                        008F  4217
                        008F  4218          SUBTEST 2
                        0092
                        0092               ;////////////////////////////////////////////////////////////////////////////////
                        0092  4219
                        0092  4220          INITIALIZE                          ;DCLO Low,init scratchpads
                        0094  4221          LDFIELD 13$,,Byte,4$,58,63          ;Load ROT Field with carry-in
                        00A5  4222          LOOP    I,1,19                      ;Setup loop
                        00AC  4223          LDFIELD 9$,I,Long,2$,31,62,LONLIT       ;Load 'A' operand
                        00BD  4224          LDFIELD 10$,I,Long,3$,31,62,LONLIT      ;Load 'B' operand
                        00CE  4225          LDFIELD 11$,I,Byte,4$,50,53         ;Load test alu function
                        00DF  4226          LOADDCS 1$,,0,<^xA>                 ;Load the test micro instrs.
                        00E6  4227          ERRLOOP                             ;Set the error loop
                        00E8  4228          FETCH   0                           ;Fetch the first micro instr.
                        00ED  4229          BRSTCLK 9                           ;Clock stop location 9
                        00F1  4230          CMPREG  WHREG,12$,I,Long            ;Check the received data
                        00FA  4231          IFERROR ,<<ALP>,<ALK>,<CLA>,<SRM>,<SRK>>        ;Continue if no error
                        0104  4232                                              ;else call out ALP,ALK,CLA
                        0104  4233          ENDLOOP I
                        0107  4234          SKIP                                ;End of test
```

```
                                010E  4235
                                010E  4236 BEGIN_TEST_DATA
                                010E
                                010E       ;-------------------------------------------------------------------------
                                010E  4237 ;+
                                010E  4238 ; The following micro words test the ALU Function of the ALP and ALK Gate
                                010E  4239 ;  Arrays.
                                010E  4240 ;-
                                010E  4241
                                010E  4242 1$:
U 00, 7700,C800,0364,0300,0470,1801  010E  4243 ;% 00   NOP                     ;NOP Micro instruction
                                0119  4247 2$:
U 01, 7700,7800,7FFF,FFFF,8470,1802  0119  4248 ;% 01   LONLIT_[0]              ;LONLIT gets 'A' Operand
U 02, 7700,4800,03C7,03D4,0470,1803  0124  4252 ;% 02   Q_LONLIT               ;Q gets 'A' operand
U 03, 7700,C860,03A4,03D8,0470,1804  012F  4256 ;% 03   M[0]_Q                 ;Save 'A' Operand
                                013A  4260 3$:
U 04, 7700,F800,7FFF,FFFF,8470,1805  013A  4261 ;% 04   LONLIT_[0]              ;LONLIT gets 'B' Operand
U 05, 7700,4800,03C7,03D4,0470,1806  0145  4265 ;% 05   Q_LONLIT               ;Q gets 'B' Operand
U 06, 7700,8840,02C7,0304,0470,1807  0150  4269 ;% 06   R[1]_Q Q_D             ;Save 'B' Operand
                                015B  4273 4$:
U 07, 7700,8800,0001,0304,0470,1808  015B  4274 ;% 07   Q_M[0]-R[1]           ;Test alu function goes here
U 08, 6700,0800,02C7,0300,0470,1809  0166  4278 ;% 08   RDM_Q Q_D             ;Read results into RDM
U 09, 7300,4800,0364,0300,0470,180A  0171  4282 ;% 09   NOP,STOP.CLOCK        ;Stop clock,Check results
                                017C  4286
                    11111111    017C  4287 5$:     .Long  ^x11111111        ;'A' OPERAND - Table 1
                    11111111    0180  4288        .Long  ^x11111111
                    41414141    0184  4289        .Long  ^x41414141
                    01010101    0188  4290        .Long  ^x01010101
                    B1B1B1B1    018C  4291        .Long  ^xB1B1B1B1
                    99999999    0190  4292        .Long  ^x99999999
                    01010101    0194  4293        .Long  ^x01010101
                    21212121    0198  4294        .Long  ^x21212121
                    12121212    019C  4295        .Long  ^x12121212
                    91919191    01A0  4296        .Long  ^x91919191
                    19191919    01A4  4297        .Long  ^x19191919
                    01010101    01A8  4298        .Long  ^x01010101
                    10101010    01AC  4299        .Long  ^x10101010
                    08080808    01B0  4300        .Long  ^x08080808
                    41414141    01B4  4301        .Long  ^x41414141
                    01010101    01B8  4302        .Long  ^x01010101
                    41414141    01BC  4303        .Long  ^x41414141
                    71717171    01C0  4304        .Long  ^x71717171
                    1B1B1B1B    01C4  4305        .Long  ^x1B1B1B1B
                    B0B0B0B0    01C8  4306        .Long  ^xB0B0B0B0
                    40404040    01CC  4307        .Long  ^x40404040
                    08080808    01D0  4308        .Long  ^x08080808
                    00808080    01D4  4309        .Long  ^x00808080
                    C8C8C8C8    01D8  4310        .Long  ^xC8C8C8C8
                    18181818    01DC  4311        .Long  ^x18181818
                    01818181    01E0  4312        .Long  ^x01818181
                    A8A8A8A8    01E4  4313        .Long  ^xA8A8A8A8
                    90909090    01E8  4314        .Long  ^x90909090
                    09090909    01EC  4315        .Long  ^x09090909
                    50505050    01F0  4316        .Long  ^x50505050
                    05050505    01F4  4317        .Long  ^x05050505
```

```
30303030  01F8  4318          .Long  ^x30303030
03030303  01FC  4319          .Long  ^x03030303
42424242  0200  4320          .Long  ^x42424242
65656565  0204  4321          .Long  ^x65656565
66666666  0208  4322          .Long  ^x66666666
84848484  020C  4323          .Long  ^x84848484
10101010  0210  4324          .Long  ^x10101010
01010101  0214  4325          .Long  ^x01010101
30303030  0218  4326          .Long  ^x30303030
03030303  021C  4327          .Long  ^x03030303
23232323  0220  4328          .Long  ^x23232323
80808080  0224  4329          .Long  ^x80808080
08080808  0228  4330          .Long  ^x08080808
30303030  022C  4331          .Long  ^x30303030
03030303  0230  4332          .Long  ^x03030303
87878787  0234  4333          .Long  ^x87878787
50505050  0238  4334          .Long  ^x50505050
05050505  023C  4335          .Long  ^x05050505
29292929  0240  4336          .Long  ^x29292929
80808080  0244  4337          .Long  ^x80808080
08080808  0248  4338          .Long  ^x08080808
00000000  024C  4339          .Long  ^x00000000
00000000  0250  4340          .Long  ^x00000000
C8C8C8C8  0254  4341          .Long  ^xC8C8C8C8
18181818  0258  4342          .Long  ^x18181818
00000000  025C  4343          .Long  ^x00000000
00000000  0260  4344          .Long  ^x00000000
D8D8D8D8  0264  4345          .Long  ^xD8D8D8D8
A0A0A0A0  0268  4346          .Long  ^xA0A0A0A0
0A0A0A0A  026C  4347          .Long  ^x0A0A0A0A
21212121  0270  4348          .Long  ^x21212121
12121212  0274  4349          .Long  ^x12121212
80808080  0278  4350          .Long  ^x80808080
08080808  027C  4351          .Long  ^x08080808
10101010  0280  4352          .Long  ^x10101010
01010101  0284  4353          .Long  ^x01010101
00000000  0288  4354          .Long  ^x00000000
00000000  028C  4355          .Long  ^x00000000
          0290  4356
08080808  0290  4357 6$:      .Long  ^x08080808          ;'B' OPERAND - Table 2
00808080  0294  4358          .Long  ^x00808080
60606060  0298  4359          .Long  ^x60606060
40404040  029C  4360          .Long  ^x40404040
C0C0C0C0  02A0  4361          .Long  ^xC0C0C0C0
08080808  02A4  4362          .Long  ^x08080808
80808080  02A8  4363          .Long  ^x80808080
62626262  02AC  4364          .Long  ^x62626262
26262606  02B0  4365          .Long  ^x26262606
68686868  02B4  4366          .Long  ^x68686868
86868606  02B8  4367          .Long  ^x86868606
88888888  02BC  4368          .Long  ^x88888888
88888808  02C0  4369          .Long  ^x88888808
00000000  02C4  4370          .Long  ^x00000000
40404040  02C8  4371          .Long  ^x40404040
10101010  02CC  4372          .Long  ^x10101010
```

```
                30303030  02D0  4373          .Long  ^x30303030
                58585858  02D4  4374          .Long  ^x58585858
                0C0C0C0C  02D8  4375          .Long  ^x0C0C0C0C
                61616161  02DC  4376          .Long  ^x61616161
                B1B1B1B1  02E0  4377          .Long  ^xB1B1B1B1
                01010101  02E4  4378          .Long  ^x01010101
                10101010  02E8  4379          .Long  ^x10101010
                C1C1C1C1  02EC  4380          .Long  ^xC1C1C1C1
                71717171  02F0  4381          .Long  ^x71717171
                17171717  02F4  4382          .Long  ^x17171717
                98989898  02F8  4383          .Long  ^x98989898
                30303030  02FC  4384          .Long  ^x30303030
                03030303  0300  4385          .Long  ^x03030303
                90909090  0304  4386          .Long  ^x90909090
                09090909  0308  4387          .Long  ^x09090909
                60606060  030C  4388          .Long  ^x60606060
                06060606  0310  4389          .Long  ^x06060606
                58585858  0314  4390          .Long  ^x58585858
                15151515  0318  4391          .Long  ^x15151515
                E4E4E4E4  031C  4392          .Long  ^xE4E4E4E4
                06060606  0320  4393          .Long  ^x06060606
                50505050  0324  4394          .Long  ^x50505050
                05050505  0328  4395          .Long  ^x05050505
                30303030  032C  4396          .Long  ^x30303030
                03030303  0330  4397          .Long  ^x03030303
                37373737  0334  4398          .Long  ^x37373737
                80808080  0338  4399          .Long  ^x80808080
                08080808  033C  4400          .Long  ^x08080808
                70707070  0340  4401          .Long  ^x70707070
                07070707  0344  4402          .Long  ^x07070707
                23232323  0348  4403          .Long  ^x23232323
                D0D0D0D0  034C  4404          .Long  ^xD0D0D0D0
                0D0D0D0D  0350  4405          .Long  ^x0D0D0D0D
                71717171  0354  4406          .Long  ^x71717171
                40404040  0358  4407          .Long  ^x40404040
                04040404  035C  4408          .Long  ^x04040404
                10101010  0360  4409          .Long  ^x10101010
                01010101  0364  4410          .Long  ^x01010101
                68686868  0368  4411          .Long  ^x68686868
                D8D8D8D8  036C  4412          .Long  ^xD8D8D8D8
                10101010  0370  4413          .Long  ^x10101010
                01010101  0374  4414          .Long  ^x01010101
                B8B8B8B8  0378  4415          .Long  ^xB8B8B8B8
                D0D0D0D0  037C  4416          .Long  ^xD0D0D0D0
                0D0D0D0D  0380  4417          .Long  ^x0D0D0D0D
                01010101  0384  4418          .Long  ^x01010101
                10101010  0388  4419          .Long  ^x10101010
                F0F0F0F0  038C  4420          .Long  ^xF0F0F0F0
                0F0F0F0F  0390  4421          .Long  ^x0F0F0F0F
                20202020  0394  4422          .Long  ^x20202020
                02020202  0398  4423          .Long  ^x02020202
                80808080  039C  4424          .Long  ^x80808080
                08080808  03A0  4425          .Long  ^x08080808
                          03A4  4426
                      00  03A4  4427 7$:       .Byte  0                ;ALU function - Table 3
```

```
                        00  03A5  4428          .Byte  0
                        00  03A6  4429          .Byte  0
                        00  03A7  4430          .Byte  0
                        01  03A8  4431          .Byte  1
                        01  03A9  4432          .Byte  1
                        01  03AA  4433          .Byte  1
                        01  03AB  4434          .Byte  1
                        01  03AC  4435          .Byte  1
                        01  03AD  4436          .Byte  1
                        01  03AE  4437          .Byte  1
                        01  03AF  4438          .Byte  1
                        01  03B0  4439          .Byte  1
                        03  03B1  4440          .Byte  3
                        02  03B2  4441          .Byte  2
                        02  03B3  4442          .Byte  2
                        02  03B4  4443          .Byte  2
                        02  03B5  4444          .Byte  2
                        01  03B6  4445          .Byte  1
                        0C  03B7  4446          .Byte  ^xC
                        0C  03B8  4447          .Byte  ^xC
                        0C  03B9  4448          .Byte  ^xC
                        0C  03BA  4449          .Byte  ^xC
                        0C  03BB  4450          .Byte  ^xC
                        0C  03BC  4451          .Byte  ^xC
                        0C  03BD  4452          .Byte  ^xC
                        04  03BE  4453          .Byte  4
                        04  03BF  4454          .Byte  4
                        04  03C0  4455          .Byte  4
                        05  03C1  4456          .Byte  5
                        05  03C2  4457          .Byte  5
                        05  03C3  4458          .Byte  5
                        05  03C4  4459          .Byte  5
                        05  03C5  4460          .Byte  5
                        05  03C6  4461          .Byte  5
                        05  03C7  4462          .Byte  5
                        05  03C8  4463          .Byte  5
                        05  03C9  4464          .Byte  5
                        05  03CA  4465          .Byte  5
                        05  03CB  4466          .Byte  5
                        05  03CC  4467          .Byte  5
                        05  03CD  4468          .Byte  5
                        05  03CE  4469          .Byte  5
                        05  03CF  4470          .Byte  5
                        05  03D0  4471          .Byte  5
                        05  03D1  4472          .Byte  5
                        05  03D2  4473          .Byte  5
                        05  03D3  4474          .Byte  5
                        05  03D4  4475          .Byte  5
                        05  03D5  4476          .Byte  5
                        05  03D6  4477          .Byte  5
                        05  03D7  4478          .Byte  5
                        06  03D8  4479          .Byte  6
                        06  03D9  4480          .Byte  6
                        07  03DA  4481          .Byte  7
                        07  03DB  4482          .Byte  7
```

```
                    08  03DC  4483          .Byte  8
                    08  03DD  4484          .Byte  8
                    0B  03DE  4485          .Byte  ^xB
                    0E  03DF  4486          .Byte  ^xE
                    0E  03E0  4487          .Byte  ^xE
                    0A  03E1  4488          .Byte  ^xA
                    0A  03E2  4489          .Byte  ^xA
                    0F  03E3  4490          .Byte  ^xF
                    0F  03E4  4491          .Byte  ^xF
                    09  03E5  4492          .Byte  9
                    09  03E6  4493          .Byte  9
                    0D  03E7  4494          .Byte  ^xD
                    0D  03E8  4495          .Byte  ^xD
                        03E9  4496
               09090909 03E9  4497 8$:      .Long  ^x09090909       ;Expected Results - table 4
               10909091 03ED  4498          .Long  ^x10909091
               E0E0E0E1 03F1  4499          .Long  ^xE0E0E0E1
               C0C0C0C1 03F5  4500          .Long  ^xC0C0C0C1
               91909090 03F9  4501          .Long  ^x91909090
               91919191 03FD  4502          .Long  ^x91919191
               21202020 0401  4503          .Long  ^x21202020
               59585858 0405  4504          .Long  ^x59585858
               86858505 0409  4505          .Long  ^x86858505
               23232323 040D  4506          .Long  ^x23232323
               33323212 0411  4507          .Long  ^x33323212
               13121212 0415  4508          .Long  ^x13121212
               22212101 0419  4509          .Long  ^x22212101
               10101010 041D  4510          .Long  ^x10101010
               00808080 0421  4511          .Long  ^x00808080
               78787878 0425  4512          .Long  ^x78787878
               08888888 0429  4513          .Long  ^x08888888
               0C8C8C8C 042D  4514          .Long  ^x0C8C8C8C
               09090909 0431  4515          .Long  ^x09090909
               B0B0B0B1 0435  4516          .Long  ^xB0B0B0B1
               71717171 0439  4517          .Long  ^x71717171
               F8F8F8F9 043D  4518          .Long  ^xF8F8F8F9
               0F8F8F90 0441  4519          .Long  ^x0F8F8F90
               F8F8F8F9 0445  4520          .Long  ^xF8F8F8F9
               59595959 0449  4521          .Long  ^x59595959
               15959596 044D  4522          .Long  ^x15959596
               41414140 0451  4523          .Long  ^x41414140
               C0C0C0C0 0455  4524          .Long  ^xC0C0C0C0
               0C0C0C0C 0459  4525          .Long  ^x0C0C0C0C
               40414141 045D  4526          .Long  ^x40414141
               14141414 0461  4527          .Long  ^x14141414
               90909090 0465  4528          .Long  ^x90909090
               09090909 0469  4529          .Long  ^x09090909
               00010101 046D  4530          .Long  ^x00010101
               80808080 0471  4531          .Long  ^x80808080
               30313131 0475  4532          .Long  ^x30313131
               90909090 0479  4533          .Long  ^x90909090
               60606060 047D  4534          .Long  ^x60606060
               06060606 0481  4535          .Long  ^x06060606
               60606060 0485  4536          .Long  ^x60606060
               06060606 0489  4537          .Long  ^x06060606
```

```
60606060  048D  4538          .Long  ^x60606060
60616161  0491  4539          .Long  ^x60616161
16161616  0495  4540          .Long  ^x16161616
00010101  0499  4541          .Long  ^x00010101
10101010  049D  4542          .Long  ^x10101010
10111111  04A1  4543          .Long  ^x10111111
80818181  04A5  4544          .Long  ^x80818181
18181818  04A9  4545          .Long  ^x18181818
00010101  04AD  4546          .Long  ^x00010101
20212121  04B1  4547          .Long  ^x20212121
12121212  04B5  4548          .Long  ^x12121212
08080808  04B9  4549          .Long  ^x08080808
00808080  04BD  4550          .Long  ^x00808080
62626260  04C1  4551          .Long  ^x62626260
E1E1E1E0  04C5  4552          .Long  ^xE1E1E1E0
00000000  04C9  4553          .Long  ^x00000000
00000000  04CD  4554          .Long  ^x00000000
31313130  04D1  4555          .Long  ^x31313130
20202020  04D5  4556          .Long  ^x20202020
02020202  04D9  4557          .Long  ^x02020202
00808080  04DD  4558          .Long  ^x00808080
08080808  04E1  4559          .Long  ^x08080808
70707070  04E5  4560          .Long  ^x70707070
07070707  04E9  4561          .Long  ^x07070707
30303030  04ED  4562          .Long  ^x30303030
03030303  04F1  4563          .Long  ^x03030303
80808080  04F5  4564          .Long  ^x80808080
08080808  04F9  4565          .Long  ^x08080808
          04FD  4566
14141414  04FD  4567 9$:      .Long  ^x14141414      ;Table 5 - 'A' Operand
10101010  0501  4568          .Long  ^x10101010
17171717  0505  4569          .Long  ^x17171717
19999999  0509  4570          .Long  ^x19999999
10101010  050D  4571          .Long  ^x10101010
10101010  0511  4572          .Long  ^x10101010
14141414  0515  4573          .Long  ^x14141414
14141414  0519  4574          .Long  ^x14141414
80808080  051D  4575          .Long  ^x80808080
0B0B0B0B  0521  4576          .Long  ^x0B0B0B0B
0C8C8C8C  0525  4577          .Long  ^x0C8C8C8C
0A8A8A8A  0529  4578          .Long  ^x0A8A8A8A
0C8C8C8C  052D  4579          .Long  ^x0C8C8C8C
01818181  0531  4580          .Long  ^x01818181
24242404  0535  4581          .Long  ^x24242404
56565606  0539  4582          .Long  ^x56565606
66666606  053D  4583          .Long  ^x66666606
48484808  0541  4584          .Long  ^x48484808
78787808  0545  4585          .Long  ^x78787808
92929202  0549  4586          .Long  ^x92929202
0D8D8D8D  054D  4587          .Long  ^x0D8D8D8D
          0551  4588
06060606  0551  4589 10$:     .Long  ^x06060606      ;'B' Operand - Table 6
04040404  0555  4590          .Long  ^x04040404
05858585  0559  4591          .Long  ^x05858585
00808080  055D  4592          .Long  ^x00808080
```

```
08080808  0561  4593          .Long  ^x08080808
01010101  0565  4594          .Long  ^x01010101
04040404  0569  4595          .Long  ^x04040404
03030303  056D  4596          .Long  ^x03030303
00000000  0571  4597          .Long  ^x00000000
16161616  0575  4598          .Long  ^x16161616
1C1C1C1C  0579  4599          .Long  ^x1C1C1C1C
09898989  057D  4600          .Long  ^x09898989
06868686  0581  4601          .Long  ^x06868686
008D8D8D  0585  4602          .Long  ^x008D8D8D
85858505  0589  4603          .Long  ^x85858505
51515101  058D  4604          .Long  ^x51515101
4E4E4E0E  0591  4605          .Long  ^x4E4E4E0E
60606000  0595  4606          .Long  ^x60606000
32323202  0599  4607          .Long  ^x32323202
17171707  059D  4608          .Long  ^x17171707
0B8B8B8B  05A1  4609          .Long  ^x0B8B8B8B
          05A5  4610
      00  05A5  4611 11$:      .Byte  0                ;ALU function - Table 7
      00  05A6  4612          .Byte  0
      02  05A7  4613          .Byte  2
      01  05A8  4614          .Byte  1
      01  05A9  4615          .Byte  1
      02  05AA  4616          .Byte  2
      02  05AB  4617          .Byte  2
      02  05AC  4618          .Byte  2
      03  05AD  4619          .Byte  3
      0C  05AE  4620          .Byte  ^xC
      0C  05AF  4621          .Byte  ^xC
      04  05B0  4622          .Byte  4
      07  05B1  4623          .Byte  7
      07  05B2  4624          .Byte  7
      05  05B3  4625          .Byte  5
      05  05B4  4626          .Byte  5
      05  05B5  4627          .Byte  5
      05  05B6  4628          .Byte  5
      05  05B7  4629          .Byte  5
      05  05B8  4630          .Byte  5
      0B  05B9  4631          .Byte  ^xB
          05BA  4632
0E0E0E0D  05BA  4633 12$:      .Long  ^x0E0E0E0D       ;Expected results - Table 8
0C0C0C0B  05BE  4634          .Long  ^x0C0C0C0B
08C8C8C8  05C2  4635          .Long  ^x08C8C8C8
18191919  05C6  4636          .Long  ^x18191919
01020202  05CA  4637          .Long  ^x01020202
07878787  05CE  4638          .Long  ^x07878787
08080807  05D2  4639          .Long  ^x08080807
08888888  05D6  4640          .Long  ^x08888888
010100FE  05DA  4641          .Long  ^x010100FE
0B0B0B0A  05DE  4642          .Long  ^x0B0B0B0A
0F8F8F8F  05E2  4643          .Long  ^x0F8F8F8F
14141414  05E6  4644          .Long  ^x14141414
26262626  05EA  4645          .Long  ^x26262626
041E1E1E  05EE  4646          .Long  ^x041E1E1E
10101010  05F2  4647          .Long  ^x10101010
```

```
                    08080808  05F6  4648          .Long   ^x08080808
                    13131313  05FA  4649          .Long   ^x13131313
                    09090909  05FE  4650          .Long   ^x09090909
                    11111111  0602  4651          .Long   ^x11111111
                    10101010  0606  4652          .Long   ^x10101010
                    04C4C4C4  060A  4653          .Long   ^x04C4C4C4
                              060E  4654
                          02  060E  4655 13$:      .Byte    2
                              060F  4656
                          00  060F  4657 14$:      .Byte    0
                              0610  4658
                              0610  4659 END_TEST_DATA
                              0610
                              0610      ;-------------------------------------------------------------------------
                              0610  4660
                              0610  4661 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
                        0026        .SBTTL  "TEST   01E     RNUM Register Data Integrity Test
                        0026  4663 ;*******************************************************************************
                        0026  4664 ;++
                        0026  4665 ;
                        0026  4666 ; FUNCTIONAL DESCRIPTION
                        0026  4667 ;
                        0026  4668 ;       This test checks the data integrity of the RNUM register by
                        0026  4669 ;       loading and reading three patterns.
                        0026  4670 ;
                        0026  4671 ; TEST ALGORITHM
                        0026  4672 ;
                        0026  4673 ;       1. Load the micro words to perform the test.
                        0026  4674 ;       2. Setup a loop of 3 iterations.
                        0026  4675 ;       3. Select and load a data pattern into LONLIT field of micro word.
                        0026  4676 ;       4. Execute the micro sequence.
                        0026  4677 ;               1. Load LONLIT Register with selected pattern.
                        0026  4678 ;               2. Load RNUM Register from LONLIT Register.
                        0026  4679 ;               3. Read RNUM Register into RDM.
                        0026  4680 ;       5. Check the results.
                        0026  4681 ;       6. Looped 3 times? If not go to Step Three.
                        0026  4682 ;
                        0026  4683 ; DATA PATTERNS
                        0026  4684 ;
                        0026  4685 ;       One of the following patterns is used each time thru the loop:
                        0026  4686 ;
                        0026  4687 ;               A
                        0026  4688 ;               5
                        0026  4689 ;               3
                        0026  4690 ;
                        0026  4691 ; LOGIC DESCRIPTION
                        0026  4692 ;
                        0026  4693 ;       This test checks the RNUM register,which is internal to the
                        0026  4694 ;       SPA Gate Array,for data lines that are SA1,SA0 or shorted together.
                        0026  4695 ;
                        0026  4696 ; ASSUMPTIONS
                        0026  4697 ;
                        0026  4698 ;       This test requires no signals from any other CPU modules but
                        0026  4699 ;       assumes an operational WBUS and RBUS.
                        0026  4700 ;
                        0026  4701 ; ERROR DESCRIPTION
                        0026  4702 ;
                        0026  4703 ;       DATA EXPECTED:Same as last loaded
                        0026  4704 ;       DATA RECEIVED:
                        0026  4705 ;       Loop Count:
                        0026  4706 ;
                        0026  4707 ;--
                        0026  4708 ;*******************************************************************************
                        0026  4709 ;///////////////////////////////////////////////////////////////////////////////
                        0026  4710
                        0026  4711         INITIALIZE                          ;DCLO Low,Init scratchpads
                        0028  4712         LOOP    I,1,3                       ;Setup loop
                        002F  4713         LDFIELD 3$,I,Byte,2$,31,34,LONLIT ;Load LONLIT field
                        0040  4714         LOADDCS 1$,,0,5                     ;Load the test micro instrs
                        0047  4715         ERRLOOP                             ;Set the error loop
                        0049  4716         FETCH   0                           ;Fetch the first micro word
```

```
                                004E  4717          BRSTCLK 4                               ;Clock stop location 4
                                0052  4718          CMPREGMSK WHREG,3$,I,4$                 ;Check the received data
                                005E  4719          IFERROR ,<<SPA>>                        ;Continue if no error else call out SPA
                                0064  4720          ENDLOOP I                               ;Continue with loop
                                0067  4721          SKIP                                    ;End of test
                                006E  4722
                                006E  4723 BEGIN_TEST_DATA
                                006E
                                006E          ;-----------------------------------------------------------------------
                                006E  4724 ;+
                                006E  4725 ; The following micro words are used to test the RNUM register.
                                006E  4726 ;-
                                006E  4727
                                006E  4728 1$:
U 00, 7700,C800,0364,0300,0470,1801  006E  4729 ;% 00    NOP                                ;NOP Micro instruction
                                0079  4733 2$:
U 01, 7700,7800,7FFF,FFFF,8470,1802  0079  4734 ;% 01    LONLIT_[0]                         ;Put data pattern here
U 02, 7700,081D,0047,03D4,0470,1803  0084  4738 ;% 02    MSRC/RNUM_WBUS,WB_LONLIT
U 03, 6700,8816,0364,0300,0470,1804  008F  4742 ;% 03    MSRC/WBUS_RNUM,RDM_WB
U 04, 7300,4800,0364,0300,0470,1805  009A  4746 ;% 04    NOP,STOP.CLOCK                     ;Stop clock and check results
                                00A5  4750
                             0A 00A5  4751 3$:      .Byte   ^xA                            ;Test pattern #1
                             05 00A6  4752          .Byte     5                            ;Test pattern #2
                             03 00A7  4753          .Byte     3                            ;Test pattern #3
                                00A8  4754
                             0F 00A8  4755 4$:      .Byte   ^xF
                                00A9  4756
                                00A9  4757 END_TEST_DATA
                                00A9
                                00A9          ;-----------------------------------------------------------------------
                                00A9  4758
                                00A9  4759 ENDTEST
```

```
0038        .SBTTL  "TEST   01F     Register Back Up Stack(RBS) and Pointer(RBSP) Tests
0038  4761 ;*******************************************************************************
0038  4762 ;++
0038  4763 ;
0038  4764 ; FUNCTIONAL DESCRIPTION
0038  4765 ;
0038  4766 ;       This test consists of nine subtests which check the Register Backup
0038  4767 ;       Stack(RBS) and the Register Backup Stack Pointer(RBSP) for operation
0038  4768 ;       and data integrity.
0038  4769 ;       The first subtest checks that the RBSP can be cleared.
0038  4770 ;       The second subtest checks that the RBSP can be incremented.
0038  4771 ;       The third test checks that the RBSP,which is a three bit counter,can
0038  4772 ;       be cleared from a count of six and the fourth subtest checks that the
0038  4773 ;       RBSP can be cleared from a count of one thus making sure that all bits
0038  4774 ;       of the counter can be reset.
0038  4775 ;       The fifth subtest checks the data integrity of the DTYPE portion of
0038  4776 ;       the RBS.
0038  4777 ;       The sixth subtest checks the RSRC portion of the RBS for data
0038  4778 ;       integrity.
0038  4779 ;       The seventh subtest checks the Decrement/Increment bit for data
0038  4780 ;       integrity.
0038  4781 ;       The eithth subtest checks the RBS for dual addressing faults.
0038  4782 ;       The nineth subtest checks that the RBSP is not cleared by RSRC
0038  4783 ;       fields other than ZERO.CLRRBSP with LIT Field equal to Zero.
0038  4784 ;       The values used for SUBTEST 9 were determined by the
0038  4785 ;       XLOG simulator and may be found in the Data Tables.
0038  4786 ;
0038  4787 ; TEST ALGORITHM
0038  4788 ;
0038  4789 ;       SUBTEST 1
0038  4790 ;
0038  4791 ;        1. Load the micro words to perform the test.
0038  4792 ;        2. Execute the micro sequence.
0038  4793 ;               1. Clear Register-Backup-Stack Pointer.
0038  4794 ;               2. Read Register-Backup-Stack Pointer into RDM.
0038  4795 ;        3. Check that Register-Backup-Stack Pointer is zero.
0038  4796 ;
0038  4797 ;       SUBTEST 2
0038  4798 ;
0038  4799 ;        1. Load the micro words to perform the test.
0038  4800 ;        2. Execute the micro sequence.
0038  4801 ;               1. Clear the Register-Backup-Stack Pointer.
0038  4802 ;        3. Setup a loop of eight iterations.
0038  4803 ;        4. Execute the micro sequence.
0038  4804 ;               1. Increment Register-Backup-Stack Pointer.
0038  4805 ;               2. Read Register-Backup-Stack Pointer into RDM.
0038  4806 ;        5. Check that Register-Backup-Stack Pointer incremented.
0038  4807 ;        6. Looped eight times?if not,go to step four.
0038  4808 ;
0038  4809 ;       SUBTEST 3
0038  4810 ;
0038  4811 ;        1. Load the micro words to perform the test.
0038  4812 ;        2. Execute the micro sequence.
0038  4813 ;               1. Clear the Register-Backup-Stack Pointer.
0038  4814 ;        3. Execute the micro sequence.
```

```
0038  4815 ;                1. Increment Register Back-up Stack Pointer six times.
0038  4816 ;                2. Clear Register Back-up Stack Pointer.
0038  4817 ;                3. Read Register Back-up Stack Pointer into RDM.
0038  4818 ;         4. Check that Register-Backup-Stack Pointer was cleared.
0038  4819 ;
0038  4820 ;        SUBTEST 4
0038  4821 ;
0038  4822 ;         1. Load the micro words to perform the test.
0038  4823 ;         2. Execute the micro sequence.
0038  4824 ;                1. Clear the Register-Backup-Stack Pointer.
0038  4825 ;         3. Execute the micro sequence.
0038  4826 ;                1. Increment Register-Backup-Stack Pointer one time.
0038  4827 ;                2. Clear Register-Backup-Stack Pointer.
0038  4828 ;                3. Read Register-Backup-Stack Pointer into RDM.
0038  4829 ;         4. Check that Register-Backup-Stack Pointer was cleared.
0038  4830 ;
0038  4831 ;        SUBTEST 5
0038  4832 ;
0038  4833 ;         1. Load the micro words to perform the test.
0038  4834 ;         2. Load a pattern from Table 6 into RSRC field of the micro word.
0038  4835 ;         3. Load a pattern from Table 8 into MSRC field of the micro word.
0038  4836 ;         4. Setup loop of three iterations.
0038  4837 ;         5. Select and load a pattern from Table 7 into DTYPE field of the
0038  4838 ;                micro word.
0038  4839 ;         6. Execute micro sequence.
0038  4840 ;                1. Clear Register-Backup-Stack Pointer.
0038  4841 ;                2. Write R Scratchpad.
0038  4842 ;                3. Clear Register-Backup-Stack Pointer.
0038  4843 ;                4. Read RBS into RDM.
0038  4844 ;         7. Check the results of the DTYPE read.
0038  4845 ;         8. Looped three times?If not go to step five.
0038  4846 ;
0038  4847 ;        SUBTEST 6
0038  4848 ;
0038  4849 ;         1. Load the micro words to perform the test.
0038  4850 ;         2. Load a pattern from Table 7 into DTYPE field of the micro word.
0038  4851 ;         3. Setup loop of four iterations.
0038  4852 ;         4. Select and load a pattern from Table 6 into RSRC field of the
0038  4853 ;                micro word.
0038  4854 ;         5. Select and load a pattern from Table 8 into MSRC field of the
0038  4855 ;                micro word.
0038  4856 ;         6. Execute micro sequence.
0038  4857 ;                1. Clear Register-Backup-Stack Pointer.
0038  4858 ;                2. Write R Scratchpad.
0038  4859 ;                3. Clear Register-Backup-Stack Pointer.
0038  4860 ;                4. Read RBS.
0038  4861 ;                5. Read RNUM into RDM.
0038  4862 ;         7. Check the results of the RNUM read.
0038  4863 ;         8. Looped four times?If not go to step four.
0038  4864 ;
0038  4865 ;        SUBTEST 7
0038  4866 ;
0038  4867 ;        1.  Load the micro words to perform the test.
0038  4868 ;        2.  Setup loop of two iterations.  When Loop I is equal to 1, perform
0038  4869 ;            an PUSH RBS+.  When Loop I is equal to 2, perform a PUSH RBS-.
```

```
0038  4870 ;       3.  Perform the following micro code to clear the Register Backup
0038  4871 ;           Stack Pointer:
0038  4872 ;               a.  Nop to get the CPU up to speed.
0038  4873 ;               b.  Do a RSRC/ZERO.CLRRBSP,LIT/0
0038  4874 ;               c.  Halt
0038  4875 ;       4.  Setup loop of six iterations to do the RBS PUSH.
0038  4876 ;       5.  Perform the following microcode to PUSH on the RBS:
0038  4877 ;               a.  Nop to get the CPU up to speed.
0038  4878 ;               b.  On loop I = 1 perform : VSIZE/1,PUSH RBS+
0038  4879 ;                   On loop I = 2 perfrom : VSIZE/1,PUSH RBS-
0038  4880 ;               c.  Halt
0038  4881 ;       6.  Continue PUSHing the RBS until loop J = 6
0038  4882 ;       7.  Execute the micro code from step 3 to clear the RBSP.
0038  4883 ;       8.  Setup loop of Six interations to do the POP RBS.
0038  4884 ;       9.  Execute the following micro code:
0038  4885 ;               a.  Nop to get the CPU up to speed.
0038  4886 ;               b.  Perform - RNUM_POPRBS MTEMP0_POPSIZ,RBS + OR - ?,NEXT/1800
0038  4887 ;               c.  Halt
0038  4888 ;       10. Compare the contents of the CSTRP register for correct results.
0038  4889 ;       11. If results don't compare, signal an error.
0038  4890 ;       12. Load micro word to perform a PUSH RBS- (VSIZE/1,PUSH RBS-).
0038  4891 ;       13. Continue with loop I until I = 2 (goto step 2 if I not equal to 2).
0038  4892 ;
0038  4893 ;       SUBTEST 8
0038  4894 ;
0038  4895 ;        1. Load the micro words to perform the test.
0038  4896 ;        2. Setup loop of eight iterations.
0038  4897 ;        3. Select and load data pattern from table 1 into RSRC field of
0038  4898 ;               micro word.
0038  4899 ;        4. Select and load data pattern from table 2 into DTYPE field
0038  4900 ;               of micro word.
0038  4901 ;        5. Select and load data pattern from table 3 into MSRC field of
0038  4902 ;               micro word.
0038  4903 ;        6. Execute micro sequence.
0038  4904 ;                    1. Clear RBS Pointer.
0038  4905 ;                    2. Write all zeroes in location zero.
0038  4906 ;                    3. Write all ones in location one.
0038  4907 ;                    4. Write all zeroes in location two.
0038  4908 ;                    5. Write all ones in location three.
0038  4909 ;                    6. Write all zeroes in location four.
0038  4910 ;                    7. Write all ones in location five.
0038  4911 ;                    8. Write all zeroes in location six.
0038  4912 ;                    9. Write all zeroes in location seven.
0038  4913 ;        7. Looped eight times?If not,go to step 3.
0038  4914 ;        8. Execute the micro sequence.
0038  4915 ;                    1. Clear the Register-Backup-Stack Pointer.
0038  4916 ;        9. Setup another loop of eight iterations.
0038  4917 ;       10. Execute micro sequence.
0038  4918 ;                    1. Read RBS into RDM.
0038  4919 ;       11. Check results.
0038  4920 ;       12.Looped eight times?If not,go to step ten.
0038  4921 ;
0038  4922 ;       SUBTEST 9
0038  4923 ;
0038  4924 ;        1. Load the micro words to perform the test.
```

```
0038  4925 ;       2. Setup a loop of seven iterations.
0038  4926 ;       3. Select values from Tables 4 and 5 and load them into RSRC field
0038  4927 ;            and LIT Field of micro word,respectively.
0038  4928 ;       4. Execute micro sequence.
0038  4929 ;            1. Init RBS Pointer to 1.
0038  4930 ;            2. Read RBS Pointer into RDM,with RSRC field and LIT Field
0038  4931 ;                 set to selected values.
0038  4932 ;       5. Check results.
0038  4933 ;       6. Looped seven times?If not,go to step three.
0038  4934 ;
0038  4935 ; DATA TABLES
0038  4936 ;
0038  4937 ;       RSRC Field      DTYPE Field     MSRC Field      RSRC Field   LIT Field
0038  4938 ;
0038  4939 ;        Table 1         Table 2         Table 3         Table 4      Table 5
0038  4940 ;
0038  4941 ;          0               0               14              17           0
0038  4942 ;          F               3               15              27           0
0038  4943 ;          0               0               14              33           0
0038  4944 ;          F               3               15              35           0
0038  4945 ;          0               0               14              36           0
0038  4946 ;          F               3               15              3F           0
0038  4947 ;          0               0               14              37           1
0038  4948 ;          F               0               14
0038  4949 ;
0038  4950 ;       RSRC Field      DTYPE Field     MSRC Field
0038  4951 ;
0038  4952 ;        Table 6         Table 7         Table 8
0038  4953 ;
0038  4954 ;          5               0               15
0038  4955 ;          A               3               14
0038  4956 ;          3               1               15
0038  4957 ;          C               2               15
0038  4958 ;
0038  4959 ; LOGIC DESCRIPTION
0038  4960 ;
0038  4961 ;       These tests check the Register Backup Stack,Register Backup
0038  4962 ;       Stack Pointer and associated circuitry,all within the SPA
0038  4963 ;       Gate Array,for proper operation.
0038  4964 ;
0038  4965 ; ASSUMPTIONS
0038  4966 ;
0038  4967 ;       This test requires no signals from any other CPU modules but
0038  4968 ;       assumes an operational WBUS and RBUS.
0038  4969 ;
0038  4970 ; ERROR DESCRIPTION
0038  4971 ;
0038  4972 ;       Data Expected:  0               ;Subtest 1
0038  4973 ;                       1               ;Subtest 2
0038  4974 ;                       2               ;Subtest 2
0038  4975 ;                       3               ;Subtest 2
0038  4976 ;                       4               ;Subtest 2
0038  4977 ;                       5               ;Subtest 2
0038  4978 ;                       6               ;Subtest 2
0038  4979 ;                       7               ;Subtest 2
```

```
                        0038  4980 ;                          0               ;Subtest 2
                        0038  4981 ;                          0               ;Subtest 3
                        0038  4982 ;                          0               ;Subtest 4
                        0038  4983 ;                          1               ;Subtest 5
                        0038  4984 ;                          2               ;Subtest 5
                        0038  4985 ;                          4               ;Subtest 5
                        0038  4986 ;                          5               ;Subtest 6
                        0038  4987 ;                          A               ;Subtest 6
                        0038  4988 ;                          3               ;Subtest 6
                        0038  4989 ;                          C               ;Subtest 6
                        0038  4990 ;                          1802            ;Subtest 7 - Loop I = 1
                        0038  4991 ;                          1800            ;Subtest 7 - Loop I = 2
                        0038  4992 ;                          0               ;Subtest 8
                        0038  4993 ;                          F               ;Subtest 8
                        0038  4994 ;                          0               ;Subtest 8
                        0038  4995 ;                          F               ;Subtest 8
                        0038  4996 ;                          0               ;Subtest 8
                        0038  4997 ;                          F               ;Subtest 8
                        0038  4998 ;                          0               ;Subtest 8
                        0038  4999 ;                          F               ;Subtest 8
                        0038  5000 ;                          1               ;Subtest 9
                        0038  5001 ;                          2               ;Subtest 9
                        0038  5002 ;                          3               ;Subtest 9
                        0038  5003 ;                          4               ;Subtest 9
                        0038  5004 ;                          5               ;Subtest 9
                        0038  5005 ;                          6               ;Subtest 9
                        0038  5006 ;                          7               ;Subtest 9
                        0038  5007 ;       Data received:
                        0038  5008 ;       Loop Count:
                        0038  5009 ;--
                        0038  5010 ;*************************************************************************
                        0038  5011
                        0038  5012         SUBTEST 1
                        003B
                        003B               ;////////////////////////////////////////////////////////////////////////
                        003B  5013
                        003B  5014         INITIALIZE                           ;DCLO Low,Init scratchpads
                        003D  5015         LOADDCS 1$,,0,4                      ;Load micro instrs for Subtest 1
                        0044  5016         ERRLOOP                              ;Set the error loop
                        0046  5017         FETCH   0                            ;Get first micro word to clear RBSP
                        004B  5018         BRSTCLK 3                            ;Clock stop location 3
                        004F  5019         CMPREGMSK WHREG,17$,,18$             ;Check results
                        005B  5020         IFERROR ,<<SPA>>                     ;Continue if no error else call out SPA
                        0061  5021
                        0061  5022         SUBTEST 2
                        0064
                        0064               ;////////////////////////////////////////////////////////////////////////
                        0064  5023
                        0064  5024         INITIALIZE                           ;DCLO Low,init scratchpads
                        0066  5025         LOADDCS 3$,,0,3                      ;Load micro instrs to clear RBSP
                        006D  5026         FETCH   0                            ;Get first micro word
                        0072  5027         BRSTCLK 2                            ;Clear RBSP
                        0076  5028         LOADDCS 2$,,0,4                      ;Load micro instrs for Subtest 2
                        007D  5029         LOOP    I,1,8                        ;Set loop count
                        0084  5030         ERRLOOP                              ;Set the error loop
```

```
                        0086  5031          FETCH   0                       ;Get first micro word
                        008B  5032          BRSTCLK 3                       ;Clock stop location 3
                        008F  5033          CMPREGMSK WHREG,20$,I,18$       ;Check results
                        009B  5034          IFERROR ,<<SPA>>                ;Continue if no error else call out SPA
                        00A1  5035          ENDLOOP I                       ;Continue with loop
                        00A4  5036
                        00A4  5037          SUBTEST 3
                        00A7
                        00A7        ;////////////////////////////////////////////////////////////////////////////////
                        00A7  5038
                        00A7  5039          INITIALIZE                      ;DCLO Low,init scratchpads
                        00A9  5040          LOADDCS 3$,,0,3                 ;Load micro instrs to clear RBSP
                        00B0  5041          FETCH   0                       ;Get first micro word
                        00B5  5042          BRSTCLK 2                       ;Clear RBSP
                        00B9  5043          LOADDCS 4$,,0,<^xA>             ;Load micro instrs for Subtest 3
                        00C0  5044          ERRLOOP                         ;Set the error loop
                        00C2  5045          FETCH   0                       ;Get  first micr word
                        00C7  5046          BRSTCLK 9                       ;Clock stop location 9
                        00CB  5047          CMPREGMSK WHREG,17$,,18$        ;Check results
                        00D7  5048          IFERROR ,<<SPA>>                ;Continue if no error else call out SPA
                        00DD  5049
                        00DD  5050          SUBTEST 4
                        00E0
                        00E0        ;////////////////////////////////////////////////////////////////////////////////
                        00E0  5051
                        00E0  5052          INITIALIZE                      ;DCLO Low,init scratchpads
                        00E2  5053          LOADDCS 3$,,0,3                 ;Load micro instrs to clear RBSP
                        00E9  5054          FETCH   0                       ;Get first micro word
                        00EE  5055          BRSTCLK 2                       ;Clear RBSP
                        00F2  5056          LOADDCS 222$,,0,5               ;Load micro instrs for Subtest 4
                        00F9  5057          ERRLOOP                         ;Set the error loop
                        00FB  5058          FETCH   0                       ;Get first micro word
                        0100  5059          BRSTCLK 4                       ;Clock stop location 4
                        0104  5060          CMPREGMSK WHREG,17$,,18$        ;Check results
                        0110  5061          IFERROR ,<<SPA>>                ;Continue if no error else call out SPA
                        0116  5062
                        0116  5063          SUBTEST 5
                        0119
                        0119        ;////////////////////////////////////////////////////////////////////////////////
                        0119  5064
                        0119  5065          INITIALIZE                      ;DCLO Low,init scratchpads
                        011B  5066          LDFIELD 21$,,Byte,6$,34,39      ;Load RSRC field
                        012C  5067          LDFIELD 23$,,Byte,6$,64,68      ;Load MSRC field
                        013D  5068          LOOP    I,1,3                   ;Set loop count
                        0144  5069          LDFIELD 22$,I,Byte,6$,40,41     ;Load DTYPE field
                        0155  5070          LOADDCS 5$,,0,6                 ;Load micro instrs for Subtest 5
                        015C  5071          ERRLOOP                         ;Set the error loop
                        015E  5072          FETCH   0                       ;Fetch the first micro word
                        0163  5073          BRSTCLK 5                       ;Clock stop location 5
                        0167  5074          CMPREGMSK WHREG,24$,I,18$       ;Check results
                        0173  5075          IFERROR ,<<SPA>>                ;Continue if no error else call out SPA
                        0179  5076          ENDLOOP I                       ;Continue with loop
                        017C  5077
                        017C  5078          SUBTEST 6
                        017F
```

```
017F        ;////////////////////////////////////////////////////////////////////////////
017F
017F                INITIALIZE                          ;DCLO Low,init scratchpads
0181                LDFIELD 22$,,Byte,8$,40,41          ;Load DTYPE Field
0192                LOOP    I,1,4                       ;Set loop count
0199                LDFIELD 21$,I,Byte,8$,34,39         ;Load RSRC Field
01AA                LDFIELD 23$,I,Byte,8$,64,68         ;Load MSRC Field
01BB                LOADDCS 7$,,0,7                     ;Load micro instrs for Subtest 6
01C2                ERRLOOP                             ;Set the error loop
01C4                FETCH   0                           ;Fetch the first micro word
01C9                BRSTCLK 6                           ;Clock stop location 6
01CD                CMPREGMSK WHREG,21$,I,18$           ;Check results
01D9                IFERROR ,<<SPA>>                    ;Continue if no error else call out SPA
01DF                ENDLOOP I                           ;Continue with loop
01E2
01E2                SUBTEST 7
01E5
01E5        ;////////////////////////////////////////////////////////////////////////////
01E5
01E5                LOADDCS         300$,,0,9           ;Load micro instrs for Subtest 7
01EC                LOOP            I,1,2               ;Set index I for two loops
01F3                ERRLOOP                             ;Loop on error from here
01F5                FETCH           0                   ;Start dcs program
01FA                BRSTCLK         2                   ;End dcs program
01FE                LOOP            J,1,6               ;Set index J for 6 loops
0205                FETCH           3                   ;Start dcs program
020A                BRSTCLK         5                   ;End dcs program
020E                ENDLOOP         J                   ;Continue until J = 6
0211                FETCH           0                   ;Start dcs program
0216                BRSTCLK         2                   ;End dcs program
021A                LOOP            J,1,6               ;Continue until J = 6
0221                FETCH           6                   ;Start dcs program
0226                BRSTCLK         8                   ;End dcs program
022A                CMPREGMSK       CSTRP,310$,I,320$,,W
0236                                                    ;Is the trapped CS address correct?
0236                IFERROR         ,<<SPA>>,<<DPM>>;Error if CS not correct
023D                ENDLOOP         J                   ;Continue until J = 6
0240                LOADDCS         305$,,4,1           ;Change micro word to do RBS-
0247                ENDLOOP         I                   ;Continue until I = 2
024A
024A                SUBTEST 8
024D
024D        ;////////////////////////////////////////////////////////////////////////////
024D
024D                INITIALIZE                          ;DCLO Low,init scratchpads
024F                LOADDCS 3$,,0,3                     ;Load micro instrs to clear RBSP
0256                FETCH   0                           ;Get first micro word
025B                BRSTCLK 2                           ;Clear RBSP
025F                LOOP    I,1,8                       ;Init loop count
0266                LDFIELD 25$,I,Byte,10$,34,39        ;Load RSRC Field
0277                LDFIELD 26$,I,Byte,10$,40,41        ;Load DTYPE Field
0288                LDFIELD 27$,I,Byte,10$,64,68        ;Load MSRC Field
0299                LOADDCS 9$,,0,3                     ;Load micro instrs
02A0                FETCH   0                           ;Get first micro word
02A5                BRSTCLK 2                           ;Clock stop location 2
```

```
                02A9  5129          ENDLOOP I                           ;Continue with loop
                02AC  5130          LOADDCS 3$,,0,3                     ;Load micro instrs to clear RBSP
                02B3  5131          FETCH   0                           ;Get first micro word
                02B8  5132          BRSTCLK 2                           ;Clear RBSP
                02BC  5133          LOOP    I,1,8                       ;Init loop count
                02C3  5134          LOADDCS 11$,,0,4                    ;Load micro instrs to read RBS
                02CA  5135          ERRLOOP                             ;Set the error loop
                02CC  5136          FETCH   0                           ;Get first micro instr
                02D1  5137          BRSTCLK 3                           ;Read Register Backup Stack
                02D5  5138          CMPREGMSK WHREG,25$,I,18$           ;Check Register Backup Stack
                02E1  5139          IFERROR ,<<SPA>>                    ;Continue if no error else call out SPA
                02E7  5140          ENDLOOP I                           ;Continue with loop
                02EA  5141
                02EA  5142          SUBTEST 9
                02ED
                02ED          ;/////////////////////////////////////////////////////////////////////////////
                02ED  5143
                02ED  5144          INITIALIZE                          ;DCLO Low,init scratchpads
                02EF  5145          LOADDCS 3$,,0,3                     ;Load micro instrs to clear RBSP
                02F6  5146          FETCH   0                           ;Get first micro word
                02FB  5147          BRSTCLK 2                           ;Clear RBSP
                02FF  5148          LOOP    I,1,7                       ;Set loop count
                0306  5149          LDFIELD 28$,I,Byte,13$,34,39        ;Load RSRC Field
                0317  5150          LDFIELD 29$,I,Byte,13$,76,77        ;Load LIT Field
                0328  5151          LOADDCS 12$,,0,4                    ;Load micro instrs
                032F  5152          ERRLOOP                             ;Set the error loop
                0331  5153          FETCH   0                           ;Get first micro instr
                0336  5154          BRSTCLK 3                           ;Read Register Backup Stack
                033A  5155          CMPREGMSK WHREG,30$,I,31$           ;Check Register Backup Stack read
                0346  5156          IFERROR ,<<SPA>>                    ;Continue if no error else call out SPA
                034C  5157          ENDLOOP I                           ;Continue with loop
                034F  5158          SKIP                                ;End of test
                0356  5159
                0356  5160 BEGIN_TEST_DATA
                0356
                0356          ;-------------------------------------------------------------------------
                0356  5161 ;+
                0356  5162 ; The following micro words are used to test the RBS and RBSP.
                0356  5163 ;-
                0356  5164
                0356  5165 1$:
U 00, 7700,C800,0364,0300,0470,1801   0356  5166 ;% 00   NOP                                 ;NOP Micro instruction
U 01, 7700,8800,0364,03DC,0470,1802   0361  5170 ;% 01   RSRC/ZERO.CLRRBSP,LIT/0
U 02, 6700,481E,0364,0300,0470,1803   036C  5174 ;% 02   MSRC/WB_RBSP,RDM_WB
U 03, 7300,C800,0364,0300,0470,1804   0377  5178 ;% 03   NOP,STOP.CLOCK                       ;NOP Micro instruction
                                      0382  5182
                                      0382  5183 2$:
U 00, 7700,C800,0364,0300,0470,1801   0382  5184 ;% 00   NOP                                 ;NOP Micro instruction
U 01, 7701,C815,0364,0200,0470,1802   038D  5188 ;% 01   MSRC/PSHADD,SIZE[LONG]
U 02, 6700,481E,0364,0300,0470,1803   0398  5192 ;% 02   MSRC/WB_RBSP,RDM_WB
U 03, 7300,C800,0364,0300,0470,1804   03A3  5196 ;% 03   NOP,STOP.CLOCK                       ;NOP Micro instruction
                                      03AE  5200
                                      03AE  5201 3$:
U 00, 7700,C800,0364,0300,0470,1801   03AE  5202 ;% 00   NOP                                 ;NOP Micro instruction
U 01, 7700,8800,0364,03DC,0470,1802   03B9  5206 ;% 01   RSRC/ZERO.CLRRBSP,LIT/0
```

```
U 02, 7300,4800,0364,0300,0470,1803  03C4  5210 ;% 02   NOP,STOP.CLOCK                    ;NOP Micro Instruction
                                     03CF  5214
                                     03CF  5215 4$:
U 00, 7700,C800,0364,0300,0470,1801  03CF  5216 ;% 00   NOP                               ;NOP Micro instruction
U 01, 7701,C815,0364,0200,0470,1802  03DA  5220 ;% 01   MSRC/PSHADD,SIZE[LONG]
U 02, 7701,4815,0364,0200,0470,1803  03E5  5224 ;% 02   MSRC/PSHADD,SIZE[LONG]
U 03, 7701,C815,0364,0200,0470,1804  03F0  5228 ;% 03   MSRC/PSHADD,SIZE[LONG]
U 04, 7701,4815,0364,0200,0470,1805  03FB  5232 ;% 04   MSRC/PSHADD,SIZE[LONG]
U 05, 7701,4815,0364,0200,0470,1806  0406  5236 ;% 05   MSRC/PSHADD,SIZE[LONG]
U 06, 7701,C815,0364,0200,0470,1807  0411  5240 ;% 06   MSRC/PSHADD,SIZE[LONG]
U 07, 7700,8800,0364,03DC,0470,1808  041C  5244 ;% 07   RSRC/ZERO.CLRRBSP,LIT/0
U 08, 6700,481E,0364,0300,0470,1809  0427  5248 ;% 08   MSRC/WB_RBSP,RDM_WB
U 09, 7300,4800,0364,0300,0470,180A  0432  5252 ;% 09   NOP,STOP.CLOCK                    ;NOP Micro instruction
                                     043D  5256
                                     043D  5257 5$:
U 00, 7700,C800,0364,0300,0470,1801  043D  5258 ;% 00   NOP                               ;NOP Micro instruction
U 01, 7700,8800,0364,03DC,0470,1802  0448  5262 ;% 01   RSRC/ZERO.CLRRBSP,LIT/0
                                     0453  5266 6$:
U 02, 7701,0820,0364,0300,0470,1803  0453  5267 ;% 02   SPW/RSIZE,VSIZE/1                 ;RSRC/21$,MSRC/23$,DTYPE/22$
U 03, 7700,8800,0364,03DC,0470,1804  045E  5271 ;% 03   RSRC/ZERO.CLRRBSP,LIT/0
U 04, 6700,081C,0364,0300,0470,1805  0469  5275 ;% 04   MSRC/READRBS,RDM_WB
U 05, 7300,4800,0364,0300,0470,1806  0474  5279 ;% 05   NOP,STOP.CLOCK                    ;NOP Micro instruction
                                     047F  5283
                                     047F  5284 7$:
U 00, 7700,C800,0364,0300,0470,1801  047F  5285 ;% 00   NOP                               ;NOP Micro instruction
U 01, 7700,8800,0364,03DC,0470,1802  048A  5289 ;% 01   RSRC/ZERO.CLRRBSP,LIT/0
                                     0495  5293 8$:
U 02, 7701,0820,0364,0300,0470,1803  0495  5294 ;% 02   SPW/RSIZE,VSIZE/1                                ;RSRC/16$,MSRC/18$,DTYPE/17$
U 03, 7700,8800,0364,03DC,0470,1804  04A0  5298 ;% 03   RSRC/ZERO.CLRRBSP,LIT/0
U 04, 7700,081C,0364,0300,0470,1805  04AB  5302 ;% 04   MSRC/READRBS
U 05, 6700,0816,0364,0300,0470,1806  04B6  5306 ;% 05   MSRC/WBUS_RNUM,RDM_WB
U 06, 7300,C800,0364,0300,0470,1807  04C1  5310 ;% 06   NOP,STOP.CLOCK
                                     04CC  5314
                                     04CC  5315 9$:
U 00, 7700,C800,0364,0300,0470,1801  04CC  5316 ;% 00   NOP                               ;NOP Micro instruction
                                     04D7  5320 10$:
U 01, 7701,8820,0364,0300,0470,1802  04D7  5321 ;% 01   SPW/RSIZE,VSIZE/1                 ;RSRC/25$,MSRC/27$,DTYPE/26$
U 02, 7300,4800,0364,0300,0470,1803  04E2  5325 ;% 02   NOP,STOP.CLOCK                    ;NOP Micro instruction
                                     04ED  5329
                                     04ED  5330 11$:
U 00, 7700,C800,0364,0300,0470,1801  04ED  5331 ;% 00   NOP                               ;NOP Micro instruction
U 01, 7700,881C,0364,0300,0470,1802  04F8  5335 ;% 01   MSRC/READRBS
U 02, 6700,0816,0364,0300,0470,1803  0503  5339 ;% 02   MSRC/WBUS_RNUM,RDM_WB
U 03, 7300,C800,0364,0300,0470,1804  050E  5343 ;% 03   NOP,STOP.CLOCK                    ;NOP Micro instruction
                                     0519  5347
                                     0519  5348 12$:
U 00, 7700,C800,0364,0300,0470,1801  0519  5349 ;% 00   NOP                               ;NOP Micro instruction
U 01, 7700,881C,0364,0300,0470,1802  0524  5353 ;% 01   MSRC/READRBS                      ;Preset Pointer
                                     052F  5357 13$:
U 02, 6700,481E,0364,0300,0470,1803  052F  5358 ;% 02   MSRC/WB_RBSP,RDM_WB              ;RSRC/28$,LIT/29$
U 03, 7300,C800,0364,0300,0470,1804  053A  5362 ;% 03   NOP,STOP.CLOCK                    ;NOP Micro instruction
                                     0545  5366
                                     0545  5367 222$:
U 00, 7700,C800,0364,0300,0470,1801  0545  5368 ;% 00   NOP                               ;NOP Micro instruction
U 01, 7701,C815,0364,0200,0470,1802  0550  5372 ;% 01   MSRC/PSHADD,SIZE[LONG]
```

```
U 02, 7700.0800.0364.03DC.0470.1803  055B  5376 ;% 02   RSRC/ZERO.CLRRBSP,LIT/0
U 03, 6700.C81E.0364.0300.0470.1804  0566  5380 ;% 03   MSRC/WB_RBSP,RDM_WB
U 04, 7300.4800.0364.0300.0470.1805  0571  5384 ;% 04   NOP,STOP.CLOCK                    ;NOP Micro instruction
                                     057C  5388
                                     057C  5389 300$:
U 00, 7700.C800.0364.0300.0470.1801  057C  5390 ;% 00   NOP
U 01, 7700.8800.0364.03DC.0470.1802  0587  5394 ;% 01   RSRC/ZERO.CLRRBSP,LIT/0
U 02, 7300.4800.0364.0300.0470.1803  0592  5398 ;% 02   STOP.CLOCK
U 03, 7700.C800.0364.0300.0470.1804  059D  5402 ;% 03   NOP
U 04, 7701.0815.0364.0300.0470.1805  05A8  5406 ;% 04   VSIZE/1,PUSH RBS+
U 05, 7300.4800.0364.0300.0470.1806  05B3  5410 ;% 05   STOP.CLOCK
U 06, 7700.C800.0364.0300.0470.1807  05BE  5414 ;% 06   NOP
U 07, 7700.087C.0364.BB00.0470.1800  05C9  5418 ;% 07   RNUM_POPRBS MTEMP0_POPSIZ,RBS + OR - ?,NEXT/1800
U 08, 7300.4800.0364.0300.0470.1809  05D4  5422 ;% 08   STOP.CLOCK
                                     05DF  5426
                                     05DF  5427 305$:
U 04, 7701.4814.0364.0300.0470.1805  05DF  5428 ;% 04   VSIZE/1,PUSH RBS-
                                     05EA  5432
                                  00 05EA  5433 17$:    .Byte   0
                                     05EB  5434
                                  0F 05EB  5435 18$:    .Byte   ^xF
                                     05EC  5436
                                  06 05EC  5437 19$:    .Byte   6
                                     05ED  5438
                                  01 05ED  5439 20$:    .Byte   1
                                  02 05EE  5440         .Byte   2
                                  03 05EF  5441         .Byte   3
                                  04 05F0  5442         .Byte   4
                                  05 05F1  5443         .Byte   5
                                  06 05F2  5444         .Byte   6
                                  07 05F3  5445         .Byte   7
                                  00 05F4  5446         .Byte   0
                                     05F5  5447
                                  05 05F5  5448 21$:    .Byte   5                       ;Table 6
                                  0A 05F6  5449         .Byte   ^xA
                                  03 05F7  5450         .Byte   3
                                  0C 05F8  5451         .Byte   ^xC
                                     05F9  5452
                                  00 05F9  5453 22$:    .Byte   0                       ;Table 7
                                  01 05FA  5454         .Byte   1
                                  02 05FB  5455         .Byte   2
                                  03 05FC  5456         .Byte   3
                                     05FD  5457
                                  15 05FD  5458 23$:    .Byte   ^x15                    ;Table 8
                                  14 05FE  5459         .Byte   ^x14
                                  15 05FF  5460         .Byte   ^x15
                                  15 0600  5461         .Byte   ^x15
                                     0601  5462
                                  01 0601  5463 24$:    .Byte   1
                                  02 0602  5464         .Byte   2
                                  04 0603  5465         .Byte   4
                                  08 0604  5466         .Byte   8
                                     0605  5467
                                  00 0605  5468 25$:    .Byte   0                       ;Table 1
                                  0F 0606  5469         .Byte   ^xF
```

```
                00  0607  5470          .Byte   0
                0F  0608  5471          .Byte   ^xF
                00  0609  5472          .Byte   0
                0F  060A  5473          .Byte   ^xF
                00  060B  5474          .Byte   0
                0F  060C  5475          .Byte   ^xF
                    060D  5476
                00  060D  5477 26$:     .Byte   0                       ;Table 2
                03  060E  5478          .Byte   3
                00  060F  5479          .Byte   0
                03  0610  5480          .Byte   3
                00  0611  5481          .Byte   0
                03  0612  5482          .Byte   3
                00  0613  5483          .Byte   0
                00  0614  5484          .Byte   0
                    0615  5485
                14  0615  5486 27$:     .Byte   ^x14                    ;Table 3
                15  0616  5487          .Byte   ^x15
                14  0617  5488          .Byte   ^x14
                15  0618  5489          .Byte   ^x15
                14  0619  5490          .Byte   ^x14
                15  061A  5491          .Byte   ^x15
                14  061B  5492          .Byte   ^x14
                14  061C  5493          .Byte   ^x14
                    061D  5494
                17  061D  5495 28$:     .Byte   ^x17                    ;Table 4
                27  061E  5496          .Byte   ^x27
                33  061F  5497          .Byte   ^x33
                35  0620  5498          .Byte   ^x35
                36  0621  5499          .Byte   ^x36
                3F  0622  5500          .Byte   ^x3F
                37  0623  5501          .Byte   ^x37
                    0624  5502
                00  0624  5503 29$:     .Byte   0                       ;Table 5
                00  0625  5504          .Byte   0
                00  0626  5505          .Byte   0
                00  0627  5506          .Byte   0
                00  0628  5507          .Byte   0
                00  0629  5508          .Byte   0
                01  062A  5509          .Byte   1
                    062B  5510
                01  062B  5511 30$:     .Byte   1
                02  062C  5512          .Byte   2
                03  062D  5513          .Byte   3
                04  062E  5514          .Byte   4
                05  062F  5515          .Byte   5
                06  0630  5516          .Byte   6
                07  0631  5517          .Byte   7
                    0632  5518
                07  0632  5519 31$:     .Byte   7
                    0633  5520
              1802  0633  5521 310$:    .WORD   ^X1802
              1800  0635  5522          .WORD   ^X1800
                    0637  5523
              3FFF  0637  5524 320$:    .WORD   ^X3FFF
```

```
                        0639  5525
                        0639  5526 END_TEST_DATA
                        0639
                        0639       ;-----------------------------------------------------------------------------
                        0639  5527
                        0639  5528 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
0023       .SBTTL  "TEST   020     Default Write to M[TEMP0] Test
0023  5530 ;*******************************************************************************
0023  5531 ;++
0023  5532 ;
0023  5533 ; FUNCTIONAL DESCRIPTION
0023  5534 ;
0023  5535 ;       This test checks that M[TEMP0] is written by default with various MSRC
0023  5536 ;       fields in the microword.
0023  5537 ;
0023  5538 ; TEST ALGORITHM
0023  5539 ;
0023  5540 ;       This test is divided into two subtests. The first subtest examines the
0023  5541 ;       default write with MSRC micro-orders that do not drive the WBUS. The
0023  5542 ;       second subtest examines the same write with MSRC micro-orders that do
0023  5543 ;       drive the WBUS.
0023  5544 ;
0023  5545 ;       The test algorithm proceeds as follows:
0023  5546 ;
0023  5547 ;       Subtest 1
0023  5548 ;
0023  5549 ;        1. Initialize the CPU.
0023  5550 ;        2. Open the window to the signature analyzer.
0023  5551 ;        3. Prepare a loop of ten iterations.
0023  5552 ;        4. Load the MSRC field of the first test micro-instruction.
0023  5553 ;        5. Load that micro-word into DCS.
0023  5554 ;        6. Execute the micro sequence:
0023  5555 ;                1. Initialize M[TEMP0] with a -1.
0023  5556 ;                2. Execute the default write.
0023  5557 ;                3. Read M[TEMP0] back to the RDM.
0023  5558 ;        7. Check the results.
0023  5559 ;        8. Looped ten times? If not go to STEP 4.
0023  5560 ;
0023  5561 ;       Subtest 2
0023  5562 ;
0023  5563 ;        9. Prepare a loop of three iterations.
0023  5564 ;       10. Load the MSRC field of the second test micro-instruction.
0023  5565 ;       11. Load that micro-word into DCS.
0023  5566 ;       12. Execute the micro sequence:
0023  5567 ;                1. Initialize M[TEMP0] with a -1.
0023  5568 ;                2. Execute the default write.
0023  5569 ;                3. Read M[TEMP0] back to the RDM.
0023  5570 ;       13. Check the results.
0023  5571 ;       14. Looped three times? If not go to STEP 4.
0023  5572 ;
0023  5573 ; DATA PATTERNS
0023  5574 ;
0023  5575 ;       The following patterns are used each time thru the loop:
0023  5576 ;
0023  5577 ;                FFFFFFFF
0023  5578 ;                00000000
0023  5579 ;
0023  5580 ;       The following are used in the MSRC Field of the micro word:
0023  5581 ;
0023  5582 ;       Subtest 1
0023  5583 ;
```

```
0023  5584 ;                    MDR                                 ; MSRC/12
0023  5585 ;                    WDR                                 ; MSRC/13
0023  5586 ;                    PSHSUB                              ; MSRC/14
0023  5587 ;                    PSHADD                              ; MSRC/15
0023  5588 ;                    RNUM_WBUS                           ; MSRC/1D
0023  5589 ;                    MA                                  ; MSRC/18
0023  5590 ;                    PCBACK                              ; MSRC/19
0023  5591 ;                    PC                                  ; MSRC/1A
0023  5592 ;                    VA                                  ; MSRC/1B
0023  5593 ;                    TB                                  ; MSRC/1F
0023  5594 ;
0023  5595 ;       Subtest 2
0023  5596 ;
0023  5597 ;                    READRBS                             ; MSRC/1C
0023  5598 ;                    WBUS_RNUM                           ; MSRC/16
0023  5599 ;                    WB_RBSP                             ; MSRC/1E
0023  5600 ;
0023  5601 ; LOGIC DESCRIPTION
0023  5602 ;
0023  5603 ;       This test checks that the SPA Gate Array will write M[TEMP0] by default
0023  5604 ;       when the MSRC field does not specify the MTEMP scratchpads.
0023  5605 ;
0023  5606 ; ASSUMPTIONS
0023  5607 ;
0023  5608 ;       This test requires no signals from any other CPU modules but assumes an
0023  5609 ;       operational MBUS, WBUS, and RBUS.
0023  5610 ;
0023  5611 ; ERROR DESCRIPTION
0023  5612 ;
0023  5613 ;       Data Expected: NOT FFFFFFFF
0023  5614 ;       Data Received:
0023  5615 ;       Loop Count:
0023  5616 ;--
0023  5617 ;*******************************************************************************
0023  5618
0023  5619          SUBTEST         1
0026
0026        ;///////////////////////////////////////////////////////////////////////////
0026  5620 ;+
0026  5621 ; This subtest will examine all MSRC micro-orders that do not drive the WBUS.
0026  5622 ;-
0026  5623          INITIALIZE                                      ; INITIALIZE THE CPU
0028  5624          LOOP            I,1,10                          ; LOOP FOR TEN MSRC FIELDS
002F  5625          LDFIELD         3$,I,BYTE,1$,64,68              ; LOAD TEST INST.'s MSRC FIELD
0040  5626          LOADDCS         1$,,4,1                         ; THEN LOAD TEST INST. INTO DCS
0047  5627          ERRLOOP                                         ; LOOP ON ERROR FROM HERE
0049  5628          FETCH           0                               ; EXECUTE AN SPW/MLONG WITH THE
004E  5629          BRSTCLK         7                               ; WBUS DRIVEN BY THE LONLIT REG
0052  5630          CMPREG          WHREG,4$,,LONG,NE               ; WAS M[TEMP0] WRITTEN?
005B  5631          IFERROR         ,<<SPA>>                        ; IF NOT, SIGNAL ERROR
0061  5632          ENDLOOP         I                               ; CONTINUE LOOP
0064  5633
0064  5634          SUBTEST         2
0067
0067        ;///////////////////////////////////////////////////////////////////////////
```

```
                                0067  5635 ;+
                                0067  5636 ; This subtest will examine all MSRC micro-orders that drive the WBUS.
                                0067  5637 ;-
                                0067  5638         LOOP            I,11,13                 ; LOOP FOR THREE MSRC FIELDS
                                006E  5639         LDFIELD         3$,I,BYTE,2$,64,68      ; LOAD TEST INST.'s MSRC FIELD
                                007F  5640         LOADDCS         2$,,4,1                 ; THEN LOAD TEST INST. INTO DCS
                                0086  5641         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                                0088  5642         FETCH           0                       ; EXECUTE AN SPW/MLONG LETTING
                                008D  5643         BRSTCLK         7                       ; MSRC DRIVE THE WBUS
                                0091  5644         CMPREG          WHREG,4$,,LONG,NE       ; WAS M[TEMP0] WRITTEN?
                                009A  5645         IFERROR         ,<<SPA>>                ; IF NOT, SIGNAL ERROR
                                00A0  5646         ENDLOOP         I                       ; CONTINUE LOOP
                                00A3  5647         SKIP                                    ; END OF TEST
                                00AA  5648
                                00AA  5649 BEGIN_TEST_DATA
                                00AA
                                00AA          ;-------------------------------------------------------------------------
                                00AA  5650 ;
                                00AA  5651 ; The following micro words are used to test the default write of M[TEMP0].
                                00AA  5652 ;
                                00AA  5653 1$:
U 04, 7700,0860,0047,03D4,0470,1805  00AA  5654 ;* 04   WB_LONLIT,SPW/MLONG               ; WRITE DEFAULT WITH LONLIT
                                00B5  5658
                                00B5  5659 2$:
U 04, 7700,4860,0364,0300,0470,1905  00B5  5660 ;* 04   SPW/MLONG                        ; WRITE DEFAULT WITH MSRC
                                00C0  5664
                             12 00C0  5665 3$:      .BYTE   ^X12                        ; MSRC/MDR
                             13 00C1  5666          .BYTE   ^X13                        ; MSRC/WDR
                             14 00C2  5667          .BYTE   ^X14                        ; MSRC/PSHSUB
                             15 00C3  5668          .BYTE   ^X15                        ; MSRC/PSHADD
                             1D 00C4  5669          .BYTE   ^X1D                        ; MSRC/RNUM_WBUS
                             18 00C5  5670          .BYTE   ^X18                        ; MSRC/MA
                             19 00C6  5671          .BYTE   ^X19                        ; MSRC/PCBACK
                             1A 00C7  5672          .BYTE   ^X1A                        ; MSRC/PC
                             1B 00C8  5673          .BYTE   ^X1B                        ; MSRC/VA
                             1F 00C9  5674          .BYTE   ^X1F                        ; MSRC/TB
                             1C 00CA  5675          .BYTE   ^X1C                        ; MSRC/READRBS
                             16 00CB  5676          .BYTE   ^X16                        ; MSRC/WBUS_RNUM
                             1E 00CC  5677          .BYTE   ^X1E                        ; MSRC/WB_RBSP
                                00CD  5678
                     FFFFFFFF   00CD  5679 4$:      .LONG   ^XFFFFFFFF                  ; DATA RESULT SHOULD NOT BE
                                00D1  5680
                                00D1  5681 BEGIN_CPU_DATA
                                0002  5682
                                0002  5683 DCS_DATA
                                0001  5684
U 00, 7700,C800,0364,0300,0470,1801  0001  5685 ;* 00   NOP                             ; GET CLOCKK UP TO SPEED
U 01, 7700,B800,0000,0000,0470,1802  000C  5689 ;* 01   LONLIT_[-1]                     ; VALUE TO INITIALIZE MTEMP0
U 02, 7700,0860,0047,03D4,0470,1803  0017  5693 ;* 02   M[TEMP0]_R[LONLIT] Q_MB        ; LOAD MTEMP0 BYPASSING ALU
U 03, 7700,7800,7FFF,FFFF,8470,1804  0022  5697 ;* 03   LONLIT_[0]                     ; RESET LONLIT
U 04, 7700,4800,0364,0300,0470,1805  002D  5701 ;* 04   NOP                             ; TEST MICRO-INSTRUCTION
U 05, 7700,4800,0057,0300,0470,1806  0038  5705 ;* 05   Q_M[TEMP0] D_R[TEMP0]          ; READ MTEMP0 TO RDM BYPASSING ALU
U 06, 6700,8800,02C7,0300,0470,1807  0043  5709 ;* 06   RDM_Q Q_D                      ; ...
U 07, 7300,C800,0364,0300,0470,1808  004E  5713 ;* 07   STOP.CLOCK                     ; HALT
                                0059  5717
```

```
                0059  5718 END_CPU_DATA
                00D1  5719
                00D1  5720 END_TEST_DATA
                00D1
                00D1       ;-----------------------------------------------------------------------------
                00D1  5721
                00D1  5722 ENDTEST
```

```
                        002A        .SBTTL  "TEST   021     MTEMP Indexed by RNUM and RNUM+1 Test
                        002A  5724 ;*******************************************************************************
                        002A  5725 ;++
                        002A  5726 ;
                        002A  5727 ; FUNCTIONAL DESCRIPTION
                        002A  5728 ;
                        002A  5729 ;       This test checks the indexing of MTEMP by RNUM and RNUM+1.Steps 1 to 5
                        002A  5730 ;       loads each MTEMP Scratchpad with its own address.Steps 6 to 10 read
                        002A  5731 ;       back using the RNUM Register as the index value.Steps 11 to 15 read
                        002A  5732 ;       them back using the RNUM Register OR 1 as the index value.
                        002A  5733 ;
                        002A  5734 ; TEST ALGORITHM
                        002A  5735 ;
                        002A  5736 ;       1. Load the micro words to perform the test.
                        002A  5737 ;       2. Setup a loop of 16 iterations.
                        002A  5738 ;       3. Select and load a register number.
                        002A  5739 ;       4. Execute the micro sequence.
                        002A  5740 ;               1. Load each MTEMP with its number.
                        002A  5741 ;       5. Looped sixteen times? If not go to STEP 3.
                        002A  5742 ;       6. Setup another loop of 16 iterations.
                        002A  5743 ;       7. Select index value.
                        002A  5744 ;       8. Execute the micro sequence.
                        002A  5745 ;               1. Load RNUM register with index value.
                        002A  5746 ;               2. Read MTEMP indexed by RNUM.
                        002A  5747 ;       9. Check the results.
                        002A  5748 ;       10.Looped sixteen times? If not go to STEP 7.
                        002A  5749 ;       11.Setup another loop of 16 iterations.
                        002A  5750 ;       12.Select index value.
                        002A  5751 ;       13.Execute the micro sequence.
                        002A  5752 ;               1. Load RNUM register with value.
                        002A  5753 ;               2. Read MTEMP indexed by RNUM+1.
                        002A  5754 ;       14.Check the results.
                        002A  5755 ;       15.Looped sixteen times? If not go to STEP 12.
                        002A  5756 ;
                        002A  5757 ; DATA PATTERNS
                        002A  5758 ;
                        002A  5759 ;       N/A
                        002A  5760 ;
                        002A  5761 ; LOGIC DESCRIPTION
                        002A  5762 ;
                        002A  5763 ;       This test checks the SPA Gate Array for proper indexing of the
                        002A  5764 ;       MTEMP scratchpads.
                        002A  5765 ;
                        002A  5766 ; ASSUMPTIONS
                        002A  5767 ;
                        002A  5768 ;       This test requires no signals from any other CPU modules but
                        002A  5769 ;       assumes an operational WBUS,RBUS and MBUS.
                        002A  5770 ;
                        002A  5771 ; ERROR DESCRIPTION
                        002A  5772 ;
                        002A  5773 ;       Data Received:Contents of each scratchpad
                        002A  5774 ;       Data Received:
                        002A  5775 ;       Loop Count:
                        002A  5776 ;--
                        002A  5777 ;*******************************************************************************
```

```
                        002A  5778
                        002A  5779          SUBTEST 1
                        002D
                        002D          ;////////////////////////////////////////////////////////////////////////////////
                        002D  5780
                        002D  5781          INITIALIZE                          ;DCLO Low,Init scratchpads
                        002F  5782          LOOP    I,1,16                      ;Setup loop
                        0036  5783          LOADREG WDREG,5$,I                  ;Load RDM D register
                        003E  5784          LDFIELD 5$,I,Byte,2$,64,68          ;Set MTEMP number
                        004F  5785          LOADDCS 1$,,0,3                     ;Load micro words
                        0056  5786          ERRLOOP                             ;Set the error loop
                        0058  5787          FETCH   0                           ;Get first micro instr
                        005D  5788          BRSTCLK 2                           ;Clock stop location 2
                        0061  5789          ENDLOOP I                           ;Continue with loop
                        0064  5790          LOOP    I,1,16                      ;Setup loop
                        006B  5791          LOADREG WDREG,5$,I                  ;Load RDM D register
                        0073  5792          LOADDCS 3$,,0,5                     ;Load micro words
                        007A  5793          ERRLOOP                             ;Set the error loop
                        007C  5794          FETCH   0                           ;Get the first micro word
                        0081  5795          BRSTCLK 4                           ;Clock stop location 4
                        0085  5796          CMPREGMSK WHREG,5$,I,7$             ;Check the received data
                        0091  5797          IFERROR ,<<SPA>>                    ;Continue if no error else call out SPA
                        0097  5798          ENDLOOP I                           ;Continue with loop
                        009A  5799
                        009A  5800          SUBTEST 2
                        009D
                        009D          ;////////////////////////////////////////////////////////////////////////////////
                        009D  5801
                        009D  5802          INITIALIZE                          ;DCLO Low,Init scratchpads
                        009F  5803          LOOP    I,1,16                      ;Setup loop
                        00A6  5804          LOADREG WDREG,5$,I                  ;Load RDM D register
                        00AE  5805          LOADDCS 4$,,0,5                     ;Load micro words
                        00B5  5806          FETCH   0                           ;Get the first micro word
                        00BA  5807          BRSTCLK 4                           ;Clock stop location 4
                        00BE  5808          CMPREGMSK WHREG,6$,I,7$             ;Check the received data
                        00CA  5809          IFERROR ,<<SPA>>                    ;Continue if no error else call out SPA
                        00D0  5810          ENDLOOP I                           ;Continue with loop
                        00D3  5811          SKIP                                ;End of test
                        00DA  5812
                        00DA  5813 BEGIN_TEST_DATA
                        00DA
                        00DA          ;-------------------------------------------------------------------------
                        00DA  5814 ;+
                        00DA  5815 ; The following micro words are used to test the MTEMP indexing.
                        00DA  5816 ;-
                        00DA  5817
                        00DA  5818 1$:
U 00, 7700,C800,0364,0300,0470,1801     00DA  5819 ;% 00    NOP                             ;NOP Micro instruction
                        00E5  5823 2$:
U 01, 5700,C860,0364,0300,0470,1802     00E5  5824 ;% 01    WB_RDM,SPW/MLONG                ;SPW/MLONG,MSRC/5$
U 02, 7300,4800,0364,0300,0470,1803     00F0  5828 ;% 02    NOP,STOP.CLOCK                  ;NOP Micro instruction
                        00FB  5832
                        00FB  5833 3$:
U 00, 7700,C800,0364,0300,0470,1801     00FB  5834 ;% 00    NOP                             ;NOP Micro instruction
U 01, 5700,C81D,0364,0300,0470,1802     0106  5838 ;% 01    MSRC/RNUM_WBUS,WB_RDM
```

```
U 02. 7700.4810.0047.0300.0470.1803  0111  5842 ;% 02   MSRC/TEMP.R,ALPCTL/WX_R.Q_M
U 03. 6700.8800.02C7.0300.0470.1804  011C  5846 ;% 03   RDM_Q_Q_D                          ;MSRC/NOP
U 04. 7300.4800.0364.0300.0470.1805  0127  5850 ;% 04   NOP,STOP.CLOCK                     ;Stop clock and check results
                                     0132  5854
                                     0132  5855 4$:
U 00. 7700.C800.0364.0300.0470.1801  0132  5856 ;% 00   NOP                               ;NOP Micro instruction
U 01. 5700.C81D.0364.0300.0470.1802  013D  5860 ;% 01   MSRC/RNUM_WBUS,WB_RDM
U 02. 7700.0811.0047.0300.0470.1803  0148  5864 ;% 02   MSRC/TEMP.R+1,ALPCTL/WX_R.Q_M
U 03. 6700.8800.02C7.0300.0470.1804  0153  5868 ;% 03   RDM_Q_Q_D                          ;MSRC/NOP
U 04. 7300.4800.0364.0300.0470.1805  015E  5872 ;% 04   NOP,STOP.CLOCK                     ;Stop clock and check results
                                     0169  5876
                                00   0169  5877 5$:     .Byte   0                       ;Table 1 and Table 2
                                     016A  5878
                                01   016A  5879 6$:     .Byte   1                       ;MTEMP Data,MTEMP addresses,RNUM Data
                                02   016B  5880         .Byte   2
                                03   016C  5881         .Byte   3
                                04   016D  5882         .Byte   4
                                05   016E  5883         .Byte   5
                                06   016F  5884         .Byte   6
                                07   0170  5885         .Byte   7
                                08   0171  5886         .Byte   8
                                09   0172  5887         .Byte   9
                                0A   0173  5888         .Byte   ^xA
                                0B   0174  5889         .Byte   ^xB
                                0C   0175  5890         .Byte   ^xC
                                0D   0176  5891         .Byte   ^xD
                                0E   0177  5892         .Byte   ^xE
                                0F   0178  5893         .Byte   ^xF
                                00   0179  5894         .Byte   0
                                     017A  5895
                                0F   017A  5896 7$:     .Byte   ^xF
                                     017B  5897
                                     017B  5898 END_TEST_DATA
                                     017B
                                     017B       ;-----------------------------------------------------------------------------
                                     017B  5899
                                     017B  5900 ENDTEST
```

```
001B       .SBTTL  "TEST   022     SPA Gate Array Test #1
001B  5902 ;*******************************************************************************
001B  5903 ;++
001B  5904 ;
001B  5905 ; FUNCTIONAL DESCRIPTION
001B  5906 ;
001B  5907 ;       This test checks that the SPA Gate Array selects only R[10]
001B  5908 ;       when writing the R Scratchpad indexed by RNUM.
001B  5909 ;
001B  5910 ; TEST ALGORITHM
001B  5911 ;
001B  5912 ;       1. Load the micro words to perform the test.
001B  5913 ;       2. Load DCS.
001B  5914 ;       3. Execute the micro sequence.
001B  5915 ;                   1. Load R[20] with 11111111.
001B  5916 ;                   2. Load R[0] with 22222222.
001B  5917 ;                   3. Load M[3] with 44444444.
001B  5918 ;                   4. Load RNUM with 0.
001B  5919 ;                   5. Load R[10] with 88888888 with RSRC field set to
001B  5920 ;                        GPR indexed by RNUM and MSRC Field set to zero.
001B  5921 ;                   6. 'OR' contents of M[3] with contents of R[10] and
001B  5922 ;                        put results back in M[3].
001B  5923 ;                   7. Load Q Register from R[0].
001B  5924 ;                   8. 'OR' contents of R[20] with contents of Q Register
001B  5925 ;                      and put results back in R[20].
001B  5926 ;                   9. 'OR' M[3] with R[20] and read results into RDM
001B  5927 ;       3. Check the results.
001B  5928 ;
001B  5929 ; DATA PATTERNS
001B  5930 ;
001B  5931 ;       The following patterns are used:
001B  5932 ;
001B  5933 ;                   00000000
001B  5934 ;                   11111111
001B  5935 ;                   22222222
001B  5936 ;                   44444444
001B  5937 ;                   88888888
001B  5938 ;
001B  5939 ; LOGIC DESCRIPTION
001B  5940 ;
001B  5941 ;       This test checks that the SPA Gate Array chip select circuitry will
001B  5942 ;       select only R[10] when the R Scratchpad is indexed by RNUM with RNUM
001B  5943 ;       equal to zero.
001B  5944 ;
001B  5945 ; SIMULATION
001B  5946 ;
001B  5947 ;       This test was developed using output from the Fault Detection
001B  5948 ;       Simulator XLOG with the following parameters:
001B  5949 ;
001B  5950 ;                   PHASE=WRITE
001B  5951 ;                   SPW=NOT MLONG
001B  5952 ;                   RNUM=0
001B  5953 ;                   MSRC=0
001B  5954 ;                   RSRC=GPR.R
001B  5955 ;
```

```
                        001B  5956 ;       Expected chip selects:
                        001B  5957 ;
                        001B  5958 ;               GPR
                        001B  5959 ;
                        001B  5960 ;       Expected M Address from SPA Gate Array:
                        001B  5961 ;
                        001B  5962 ;               3
                        001B  5963 ;
                        001B  5964 ;       Expected R Address from SPA Gate Array:
                        001B  5965 ;
                        001B  5966 ;               0
                        001B  5967 ;
                        001B  5968 ; ASSUMPTIONS
                        001B  5969 ;
                        001B  5970 ;       This test requires no signals from any other CPU modules.
                        001B  5971 ;       It assumes thet previous tests hve been run to verify that
                        001B  5972 ;       the logic and registers used by this test are operational.
                        001B  5973 ;
                        001B  5974 ; ERROR DESCRIPTION
                        001B  5975 ;
                        001B  5976 ;       Data Expected:FFFFFFFF
                        001B  5977 ;       Data Received:
                        001B  5978 ;
                        001B  5979 ;--
                        001B  5980 ;*******************************************************************************
                        001B  5981 ;///////////////////////////////////////////////////////////////////////////////
                        001B  5982
                        001B  5983         INITIALIZE                          ;DCLO Low,Init scratchpads
                        001D  5984         LOADDCS 1$,,0,<^x15>                ;Load the test micro instrs
                        0024  5985         ERRLOOP                             ;Set the error loop
                        0026  5986         FETCH   0                           ;Fetch the first micro word
                        002B  5987         BRSTCLK <^x14>                      ;Clock stop location 14
                        002F  5988         CMPREG  WHREG,2$,,Long              ;Check the received data
                        0038  5989         IFERROR ,<<SPA>>                    ;Continue if no error else call out SPA
                        003E  5990         SKIP                                ;End of test
                        0045  5991
                        0045  5992 BEGIN_TEST_DATA
                        0045
                        0045          ;-------------------------------------------------------------------------
                        0045  5993 ;+
                        0045  5994 ; The following micro words are used to test the SPA Gate Array.
                        0045  5995 ;-
                        0045  5996
                        0045  5997 1$:
U 00, 7700,C800,0364,0300,0470,1801     0045  5998 ;% 00    NOP                                 ;NOP Micro instruction
U 01, 7700,7800,7777,7777,0470,1802     0050  6002 ;% 01    LONLIT_[011111111]
U 02, 7700,4800,03C7,03D4,0470,1803     005B  6006 ;% 02    Q_LONLIT
U 03, 7700,8840,02C7,0380,0470,1804     0066  6010 ;% 03    R[20]_Q Q_D                         ;Init IPR0
U 04, 7700,F800,6EEE,EEEE,8470,1805     0071  6014 ;% 04    LONLIT_[022222222]
U 05, 7700,4800,03C7,03D4,0470,1806     007C  6018 ;% 05    Q_LONLIT
U 06, 7700,C840,02C7,0300,0470,1807     0087  6022 ;% 06    R[0]_Q Q_D                          ;Init RTEMP0
U 07, 7700,B800,5DDD,DDDD,8470,1808     0092  6026 ;% 07    LONLIT_[044444444]
U 08, 7700,4863,0247,03D4,0470,1809     009D  6030 ;% 08    M[3]_LONLIT                         ;Init MTEMP3
U 09, 7700,F800,7FFF,FFFF,8470,180A     00A8  6034 ;% 09    LONLIT_[0]                          ;Clear LONLIT
U 0A, 7700,C81D,0057,03D4,0470,180B     00B3  6038 ;% 0A    RNUM_LONLIT                         ;Set RNUM to Zero
```

```
U 0B. 7700.4800.03C7.03D4.0470.180C  00BE  6042 ;% 0B   Q_LONLIT                          ;Clear Q Register
U 0C. 7700.8840.02C7.0340.0470.180D  00C9  6046 ;% 0C   R[10]_Q Q_D                       ;Init GPR0
U 0D. 7700.7800.3BBB.BBBB.8470.180E  00D4  6050 ;% 0D   LONLIT_[0888888888]
U 0E. 7700.4800.03C7.03D4.0470.180F  00DF  6054 ;% 0E   Q_LONLIT
U 0F. 7700.8840.02C7.0340.0470.1810  00EA  6058 ;% 0F   R[10]_Q Q_D                       ;RSRC/GPR.R
U 10. 7700.4863.0024.0340.0470.1811  00F5  6062 ;% 10   M[3]_MB.OR.R[10]                 ;'OR' MTEMP0 and GPR0 and
                                     0100  6066                                        ;put results in MTEMP3 (33..)
U 11. 7700.0800.5BE5.0300.0470.1812  0100  6067 ;% 11   Q_R[0]
U 12. 7700.C840.03A4.0380.0470.1813  010B  6071 ;% 12   R[20]_RB.OR.Q                    ;'OR' IPR0 with RTEMP0 and
                                     0116  6075                                        ;put results in IPR0 (CC..)
U 13. 6700.4803.0024.0380.0470.1814  0116  6076 ;% 13   RDM_M[3].OR.R[20]               ;'OR' IPR0 with MTEMP3 and
                                     0121  6080                                        ;put results in RDM (FF..)
U 14. 7300.C800.0364.0300.0470.1815  0121  6081 ;% 14   NOP,STOP.CLOCK                   ;Stop clock and check results
                                     012C  6085
                         FFFFFFFF    012C  6086 2$:     .Long   ^xFFFFFFFF             ;Expected data
                                     0130  6087
                                     0130  6088 END_TEST_DATA
                                     0130
                                     0130       ;-------------------------------------------------------------------------
                                     0130  6089
                                     0130  6090 ENDTEST
```

```
001B          .SBTTL  "TEST   023     SPA Gate Array Test #2
001B  6092 ;*******************************************************************************
001B  6093 ;++
001B  6094 ;
001B  6095 ; FUNCTIONAL DESCRIPTION
001B  6096 ;
001B  6097 ;       This test checks that the SPA Gate Array chip select circuitry
001B  6098 ;       functions properly when doing an R Scratchpad write with the
001B  6099 ;       MSRC field set to TEMP8.
001B  6100 ;
001B  6101 ; TEST ALGORITHM
001B  6102 ;
001B  6103 ;       1. Load the micro words to perform the test.
001B  6104 ;       2. Execute the micro sequence.
001B  6105 ;                1. Load RNUM Register with 0.
001B  6106 ;                2. Load Register R[10] with 11111111.
001B  6107 ;                3. Load Register R[20] with 22222222.
001B  6108 ;                4. Load Register M[00] with 88888888.
001B  6109 ;                5. Load Register R[00] with 88888888 while MSRC field
001B  6110 ;                   is set to TEMP8.
001B  6111 ;                6. 'AND' Register M[0] and Register R[0] and put results in
001B  6112 ;                   Q Register.
001B  6113 ;                7. 'OR' Register R[10] with Q Register and put results in Q
001B  6114 ;                   Register.
001B  6115 ;                8.'OR' Register R[20] with Q Register and put results in RDM.
001B  6116 ;
001B  6117 ;       3. Check the results.
001B  6118 ;
001B  6119 ; DATA PATTERNS
001B  6120 ;
001B  6121 ;       The following patterns are used:
001B  6122 ;
001B  6123 ;       00000000
001B  6124 ;       11111111
001B  6125 ;       22222222
001B  6126 ;       44444444
001B  6127 ;       88888888
001B  6128 ;
001B  6129 ; SIMULATION
001B  6130 ;
001B  6131 ;       This test was developed using output from the Fault Detection
001B  6132 ;       Simulator XLOG with the following parameters:
001B  6133 ;
001B  6134 ;               PHASE=WRITE
001B  6135 ;               SPW=NOT MLONG
001B  6136 ;               RNUM=0
001B  6137 ;               MSRC=TEMP8
001B  6138 ;               RSRC=0
001B  6139 ;
001B  6140 ;       Expected chip selects:
001B  6141 ;
001B  6142 ;               RTEMP,MTEMP
001B  6143 ;
001B  6144 ;       Expected MSPA Address from SPA Gate Array:
001B  6145 ;
```

```
                        001B  6146 ;               0
                        001B  6147 ;
                        001B  6148 ;       Expected RSPA Address from SPA Gate Array:
                        001B  6149 ;
                        001B  6150 ;               0
                        001B  6151 ;
                        001B  6152 ; LOGIC DESCRIPTION
                        001B  6153 ;
                        001B  6154 ;       This test checks that the SPA Gate Array asserts only
                        001B  6155 ;       RCS TMP L and MCS TMP L chip selects during an R Scratchpad
                        001B  6156 ;       write with the MSRC field set to TEMP8.
                        001B  6157 ;
                        001B  6158 ; ASSUMPTIONS
                        001B  6159 ;
                        001B  6160 ;       This test requires no signals from any other CPU modules.
                        001B  6161 ;       It assumes that previous tests have been run to verify that
                        001B  6162 ;       the logic and registers used by this test are operational.
                        001B  6163 ;
                        001B  6164 ; ERROR DESCRIPTION
                        001B  6165 ;
                        001B  6166 ;       DATA EXPECTED:BBBBBBBB
                        001B  6167 ;       DATA RECEIVED:
                        001B  6168 ;
                        001B  6169 ;--
                        001B  6170 ;*******************************************************************************
                        001B  6171 ;///////////////////////////////////////////////////////////////////////////////
                        001B  6172
                        001B  6173         INITIALIZE                          ;DCLO Low,Init scratchpads
                        001D  6174         LOADDCS 1$,,0,<^x11>                ;Load the test micro instrs
                        0024  6175         ERRLOOP                             ;Set the error loop
                        0026  6176         FETCH   0                           ;Fetch the first micro word
                        002B  6177         BRSTCLK <^x10>                      ;Clock stop location 10
                        002F  6178         CMPREG  WHREG,2$,,Long              ;Check the received data
                        0038  6179         IFERROR ,<<SPA>>                    ;Continue if no error else call out SPA
                        003E  6180         SKIP                                ;End of test
                        0045  6181
                        0045  6182 BEGIN_TEST_DATA
                        0045
                        0045          ;-------------------------------------------------------------------------
                        0045  6183 ;+
                        0045  6184 ; The following micro words are used to test the SPA Gate Array.
                        0045  6185 ;-
                        0045  6186
                        0045  6187 1$:
U 00, 7700,C800,0364,0300,0470,1801     0045  6188 ;% 00    NOP                                 ;NOP Micro instruction
U 01, 7700,7800,7FFF,FFFF,8470,1802     0050  6192 ;% 01    LONLIT_[0]
U 02, 7700,081D,0047,03D4,0470,1803     005B  6196 ;% 02    MSRC/RNUM_WBUS,WB_LONLIT
U 03, 7700,7800,7777,7777,0470,1804     0066  6200 ;% 03    LONLIT_[11111111]
U 04, 7700,4800,03C7,03D4,0470,1805     0071  6204 ;% 04    Q_LONLIT
U 05, 7700,0840,02C7,0340,0470,1806     007C  6208 ;% 05    R[10]_Q Q_D
U 06, 7700,7800,6EEE,EEEE,8470,1807     0087  6212 ;% 06    LONLIT_[22222222]
U 07, 7700,C800,03C7,03D4,0470,1808     0092  6216 ;% 07    Q_LONLIT
U 08, 7700,0840,02C7,0380,0470,1809     009D  6220 ;% 08    R[20]_Q Q_D
U 09, 7700,F800,3BBB,BBBB,8470,180A     00A8  6224 ;% 09    LONLIT_[88888888]
U 0A, 7700,C800,03C7,03D4,0470,180B     00B3  6228 ;% 0A    Q_LONLIT
```

```
U 0B, 7700,0848,02C7,0300,0470,180C  00BE  6232 ;% 0B   R[0]_Q Q_D,MSRC/TEMP8          ;MSRC/TEMP8
U 0C, 7700,8800,0021,0300,0470,180D  00C9  6236 ;% 0C   Q_M[0].AND.R[0]               ;Should be 88888888
U 0D, 7700,C800,03A5,0340,0470,180E  00D4  6240 ;% 0D   Q_R[10].OR.Q                 ;MUX/R.Q,DQ/Q<-W   Should be 99999999
U 0E, 7700,4800,03A5,0380,0470,180F  00DF  6244 ;% 0E   Q_R[20].OR.Q                 ;MUX/R.Q,DQ/Q<-W   Should be BBBBBBBB
U 0F, 6700,8800,02C7,0300,0470,1810  00EA  6248 ;% 0F   RDM_Q Q_D
U 10, 7300,4800,0364,0300,0470,1811  00F5  6252 ;% 10   NOP,STOP.CLOCK               ;Stop clock and check results
                                     0100  6256
                           BBBBBBBB  0100  6257 2$:     .Long   ^xBBBBBBBB          ;Results go here
                                     0104  6258
                                     0104  6259 END_TEST_DATA
                                     0104
                                     0104       ;-----------------------------------------------------------------------
                                     0104  6260
                                     0104  6261 ENDTEST
```

```
                        001B        .SBTTL  "TEST   024     SPA Gate Array Test #3
                        001B  6263 ;*******************************************************************************
                        001B  6264 ;++
                        001B  6265 ;
                        001B  6266 ; FUNCTIONAL DESCRIPTION
                        001B  6267 ;
                        001B  6268 ;       This test checks that the SPA Gate Array asserts the correct
                        001B  6269 ;       chip selects during an MTMP scratchpad write with the RSRC
                        001B  6270 ;       field set to various selected values.
                        001B  6271 ;
                        001B  6272 ; TEST ALGORITHM
                        001B  6273 ;
                        001B  6274 ;       1. Load the micro words to perform the test.
                        001B  6275 ;       2. Setup a loop of 6 iterations.
                        001B  6276 ;       3. Select and load RSRC field into micro word.
                        001B  6277 ;       4. Execute the micro sequence.
                        001B  6278 ;               1. Load RNUM Register and M[0] Register with 0.
                        001B  6279 ;               2. Load R[20] Register with 88888888.
                        001B  6280 ;               3. Load R[10] Register with 44444444.
                        001B  6281 ;               4. Load M[0] Register with 33333333 with selected RSRC
                        001B  6282 ;                   field in micro word.
                        001B  6283 ;               5. 'AND' Register M[0] and Register R[0] and put results in
                        001B  6284 ;                  Q Register.
                        001B  6285 ;               6. 'OR' Register R[10] with Q Register and put results in Q
                        001B  6286 ;                   Register.
                        001B  6287 ;               7.'OR' Register R[20] with Q Register and put results in RDM.
                        001B  6288 ;       5. Check the results.
                        001B  6289 ;       6. Looped 6 times? If not go to STEP 3.
                        001B  6290 ;
                        001B  6291 ; DATA PATTERNS
                        001B  6292 ;
                        001B  6293 ;       The following patterns are used each time thru the loop:
                        001B  6294 ;
                        001B  6295 ;       33333333
                        001B  6296 ;       44444444
                        001B  6297 ;       88888888
                        001B  6298 ;
                        001B  6299 ; SIMULATION
                        001B  6300 ;
                        001B  6301 ;       This test was developed using output from the Fault Detection
                        001B  6302 ;       Simulator XLOG with the following parameters:
                        001B  6303 ;
                        001B  6304 ;               PHASE=WRITE
                        001B  6305 ;               SPW=MLONG
                        001B  6306 ;               RNUM=0
                        001B  6307 ;               MSRC=0
                        001B  6308 ;               RSRC=11,38,33,18,24,12,(Hex)
                        001B  6309 ;
                        001B  6310 ;       Expected chip selects:
                        001B  6311 ;
                        001B  6312 ;               RTEMP,MTEMP
                        001B  6313 ;
                        001B  6314 ;       Expected MSPA Address from SPA Gate Array:
                        001B  6315 ;
                        001B  6316 ;               0
```

```
                        001B  6317 ;
                        001B  6318 ;       Expected RSPA Address from SPA Gate Array:
                        001B  6319 ;
                        001B  6320 ;                 0
                        001B  6321 ;
                        001B  6322 ; LOGIC DESCRIPTION
                        001B  6323 ;
                        001B  6324 ;       This test checks that the SPA Gate Array asserts only
                        001B  6325 ;       RCS TMP L or RCS IPR L and MCS TMP L chip selects during an
                        001B  6326 ;       M Scratchpad write with various values in the RSRC field.
                        001B  6327 ;
                        001B  6328 ; ASSUMPTIONS
                        001B  6329 ;
                        001B  6330 ;       This test requires no signals from any other CPU modules.
                        001B  6331 ;       It assumes that previous tests have been run to verify that
                        001B  6332 ;       the logic and registers used by this test are operational.
                        001B  6333 ;
                        001B  6334 ; ERROR DESCRIPTION
                        001B  6335 ;
                        001B  6336 ;       Data Expected:FFFFFFFF
                        001B  6337 ;       Data Received:
                        001B  6338 ;       Loop Count:
                        001B  6339 ;
                        001B  6340 ;--
                        001B  6341 ;*******************************************************************************
                        001B  6342 ;///////////////////////////////////////////////////////////////////////////////
                        001B  6343
                        001B  6344         INITIALIZE                          ;DCLO Low,Init scratchpads
                        001D  6345         LOOP    I,1,6                       ;Setup loop
                        0024  6346         LDFIELD 3$,I,Byte,2$,34,39          ;Load micro word field
                        0035  6347         LOADDCS 1$,,0,<^x11>                ;Load the test micro instrs
                        003C  6348         ERRLOOP                             ;Set the error loop
                        003E  6349         FETCH   0                           ;Fetch the first micro word
                        0043  6350         BRSTCLK <^x10>                      ;Clock stop location 16
                        0047  6351         CMPREG  WHREG,4$,,Long              ;Check the received data
                        0050  6352         IFERROR ,<<SPA>>                    ;Continue if no error else call out SPA
                        0056  6353         ENDLOOP I                           ;Continue with loop
                        0059  6354         SKIP                                ;End of test
                        0060  6355
                        0060  6356 BEGIN_TEST_DATA
                        0060
                        0060               ;-------------------------------------------------------------------
                        0060  6357 ;+
                        0060  6358 ; The following micro words are used to test the SPA Gate Array.
                        0060  6359 ;-
                        0060  6360
                        0060  6361 1$:
U 00, 7700,C800,0364,0300,0470,1801     0060  6362 ;% 00    NOP                                 ;NOP Micro instruction
U 01, 7700,7800,7FFF,FFFF,8470,1802     006B  6366 ;% 01    LONLIT_[00000000]
U 02, 7700,481D,03C7,03D4,0470,1803     0076  6370 ;% 02    Q_LONLIT,MSRC/RNUM_WBUS             ;MSRC/RNUM_WBUS
U 03, 7700,C860,03A4,03D8,0470,1804     0081  6374 ;% 03    M[0]_Q                              ;Init M[0]
U 04, 7700,F800,3BBB,BBBB,8470,1805     008C  6378 ;% 04    LONLIT_[88888888]
U 05, 7700,4800,03C7,03D4,0470,1806     0097  6382 ;% 05    Q_LONLIT
U 06, 7700,8840,02C7,0380,0470,1807     00A2  6386 ;% 06    R[20]_Q Q_D                         ;Load R[20]
U 07, 7700,B800,5DDD,DDDD,8470,1808     00AD  6390 ;% 07    LONLIT_[44444444]
```

```
U 08, 7700,4800,03C7,03D4,0470,1809  00B8  6394 ;% 08   Q_LONLIT
U 09, 7700,0840,02C7,0340,0470,180A  00C3  6398 ;% 09   R[10]_Q Q_D                    ;Load R[10]
U 0A, 7700,7800,6666,6666,0470,180B  00CE  6402 ;% 0A   LONLIT_[33333333]
U 0B, 7700,4800,03C7,03D4,0470,180C  00D9  6406 ;% 0B   Q_LONLIT
                                     00E4  6410 2$:
U 0C, 7700,C860,03A4,03D8,0470,180D  00E4  6411 ;% 0C   M[0]_Q                       ;RSRC/3$
U 0D, 7700,8800,0021,0300,0470,180E  00EF  6415 ;% 0D   Q_M[0].AND.R[0]
U 0E, 7700,4800,03A5,0340,0470,180F  00FA  6419 ;% 0E   Q_R[10].OR.Q                 ;MUX/R.Q,DQ/Q<-W
U 0F, 6700,C800,03A5,0380,0470,1810  0105  6423 ;% 0F   RDM_WB,Q_R[20].OR.Q          ;MUX/R.Q,DQ/Q<-W
U 10, 7300,4800,0364,0300,0470,1811  0110  6427 ;% 10   NOP,STOP.CLOCK               ;Stop clock and check results
                                     011B  6431
                                     011B  6432 ; RSRC Fields
                                     011B  6433
                                  11 011B  6434 3$:      .Byte   ^x11               ;R[1]
                                  38 011C  6435          .Byte   ^x38               ;TEMP.ROR1
                                  33 011D  6436          .Byte   ^x33               ;GPR.R
                                  18 011E  6437          .Byte   ^x18               ;R[18]
                                  24 011F  6438          .Byte   ^x24               ;R[24]
                                  12 0120  6439          .Byte   ^x12               ;R[12]
                                     0121  6440
                            FFFFFFFF 0121  6441 4$:      .Long   ^xFFFFFFFF        ;Test results
                                     0125  6442
                                     0125  6443 END_TEST_DATA
                                     0125
                                     0125       ;-------------------------------------------------------------------------
                                     0125  6444
                                     0125  6445 ENDTEST
```

```
                        001B        .SBTTL  "TEST   025     SPA Gate Array Test #4
                        001B  6447 ;*******************************************************************************
                        001B  6448 ;++
                        001B  6449 ;
                        001B  6450 ; FUNCTIONAL DESCRIPTION
                        001B  6451 ;
                        001B  6452 ;       This test checks the SPA Gate Array for proper operation of
                        001B  6453 ;       the chip selects when indexing by the RNUM Register during
                        001B  6454 ;       a Read R Scratchpad operation.
                        001B  6455 ;
                        001B  6456 ; TEST ALGORITHM
                        001B  6457 ;
                        001B  6458 ;       1. Load the micro words to perform the test.
                        001B  6459 ;       2. Setup a loop of 5 iterations.
                        001B  6460 ;       3. Select and load the fields into the micro word.
                        001B  6461 ;       4. Execute the micro sequence.
                        001B  6462 ;               1. Load Register,selected from first table,with
                        001B  6463 ;                   11111111.
                        001B  6464 ;               2. Load Register,selected from second table,with
                        001B  6465 ;                   22222222.
                        001B  6466 ;               3. Load Register,selected from third table,with
                        001B  6467 ;                   44444444.
                        001B  6468 ;               4. Load M[0] Register with 8888888 pattern.
                        001B  6469 ;               5. Load RNUM Register with value selected from fourth table.
                        001B  6470 ;               6. 'AND' R Scratchpad referenced by RSRC field value selected
                        001B  6471 ;                   from fifth table with M[0] and  put results in Q
                        001B  6472 ;                   Register.
                        001B  6473 ;               7. 'OR' Q Register with R Scratchpad selected from second
                        001B  6474 ;                   table and put results back in Q Register.
                        001B  6475 ;               8. 'OR' Q Register with R Scratchpad selected from first
                        001B  6476 ;                   table and put results RDM.
                        001B  6477 ;       5. Check the results.
                        001B  6478 ;       6. Looped 5 times? If not go to STEP 3.
                        001B  6479 ;
                        001B  6480 ; DATA PATTERNS
                        001B  6481 ;
                        001B  6482 ;       The following patterns are used for all tests:
                        001B  6483 ;
                        001B  6484 ;               11111111
                        001B  6485 ;               22222222
                        001B  6486 ;               44444444
                        001B  6487 ;               88888888
                        001B  6488 ;
                        001B  6489 ; DATA TABLES
                        001B  6490 ;
                        001B  6491 ;               R[14]                   ;INIT DATA-TABLE 1
                        001B  6492 ;               R[11]
                        001B  6493 ;               R[19]
                        001B  6494 ;               R[13]
                        001B  6495 ;               R[1C]
                        001B  6496 ;
                        001B  6497 ;               R[24]                   ;INIT DATA-TABLE 2
                        001B  6498 ;               R[21]
                        001B  6499 ;               R[29]
                        001B  6500 ;               R[23]
```

```
001B  6501 ;                    R[2C]
001B  6502 ;
001B  6503 ;                    R[4]                    ;INIT DATA-TABLE 3
001B  6504 ;                    R[1]
001B  6505 ;                    R[9]
001B  6506 ;                    R[3[
001B  6507 ;                    R[C]
001B  6508 ;
001B  6509 ;                    4                       ;RNUM DATA-TABLE 4
001B  6510 ;                    1
001B  6511 ;                    8
001B  6512 ;                    3
001B  6513 ;                    1
001B  6514 ;
001B  6515 ;                    TEMP.R                  ;RSRC MICROWORD READ DATA-TABLE 5
001B  6516 ;                    TEMP.R
001B  6517 ;                    TEMP.ROR.1
001B  6518 ;                    TEMP.ROR.1
001B  6519 ;                    R[2C]
001B  6520 ;
001B  6521 ; SIMULATION
001B  6522 ;
001B  6523 ;       This test was developed using output from the Fault Detection
001B  6524 ;       Simulator XLOG with the following parameters:
001B  6525 ;
001B  6526 ;               PHASE=READ
001B  6527 ;               SPW=NOT MLONG
001B  6528 ;               RNUM=4,1,8,3,1
001B  6529 ;               MSRC=0
001B  6530 ;               RSRC=TEMP.R,TEMP.R,TEMP.ROR1,TEMP.ROR1,R[2C]
001B  6531 ;
001B  6532 ;       Expected chip selects:
001B  6533 ;
001B  6534 ;               RTEMP,MTEMP
001B  6535 ;               RTEMP,MTEMP
001B  6536 ;               RTEMP,MTEMP
001B  6537 ;               RTEMP,MTEMP
001B  6538 ;                IPR,MTEMP
001B  6539 ;
001B  6540 ;       Expected MSPA Address from SPA Gate Array:
001B  6541 ;
001B  6542 ;               0
001B  6543 ;
001B  6544 ;       Expected RSPA Address from SPA Gate Array:
001B  6545 ;
001B  6546 ;               4
001B  6547 ;               1
001B  6548 ;               9
001B  6549 ;               3
001B  6550 ;               C
001B  6551 ;
001B  6552 ; LOGIC DESCRIPTION
001B  6553 ;
001B  6554 ;       This test checks that the SPA Gate Array asserts the proper
001B  6555 ;       chip selects.
```

```
                        001B  6556 ;
                        001B  6557 ; ASSUMPTIONS
                        001B  6558 ;
                        001B  6559 ;       This test requires no signals from any other CPU modules.
                        001B  6560 ;       It assumes that previous tests have been run to verify that
                        001B  6561 ;       the logic and the registers used by this test are opeational
                        001B  6562 ;
                        001B  6563 ; ERROR DESCRIPTION
                        001B  6564 ;
                        001B  6565 ;       Data Expected:FFFFFFFF
                        001B  6566 ;       Data Received:
                        001B  6567 ;       Loop Count:
                        001B  6568 ;
                        001B  6569 ;--
                        001B  6570 ;*******************************************************************************
                        001B  6571 ;///////////////////////////////////////////////////////////////////////////////
                        001B  6572
                        001B  6573         INITIALIZE                          ;DCLO Low,Init scratchpads
                        001D  6574         LOOP    I,1,5                       ;Setup loop
                        0024  6575         LDFIELD 12$,I,Byte,2$,34,39         ;Load micro word field
                        0035  6576         LDFIELD 13$,I,Byte,3$,34,39         ;Load micro word field
                        0046  6577         LDFIELD 14$,I,Byte,4$,34,39         ;Load micro word field
                        0057  6578         LDFIELD 15$,I,Byte,5$,31,62,LONGLIT;Load micro word field
                        0068  6579         LDFIELD 16$,I,Byte,6$,34,39         ;Load micro word field
                        0079  6580         LDFIELD 13$,I,Byte,7$,34,39         ;Load micro word field
                        008A  6581         LDFIELD 12$,I,Byte,8$,34,39         ;Load micro word field
                        009B  6582         LOADDCS 1$,,0,<^x13>                ;Load the test micro instrs
                        00A2  6583         ERRLOOP                             ;Set the error loop
                        00A4  6584         FETCH   0                           ;Fetch the first micro word
                        00A9  6585         BRSTCLK <^x12>                      ;Clock stop location 11
                        00AD  6586         CMPREG  WHREG,17$,I,Long            ;Check the received data
                        00B6  6587         IFERROR ,<<SPA>>                    ;Continue if no error else call out SPA
                        00BC  6588         ENDLOOP I                           ;Continue with loop
                        00BF  6589         SKIP                                ;End of test
                        00C6  6590
                        00C6  6591 BEGIN_TEST_DATA
                        00C6
                        00C6               ;-------------------------------------------------------------------------
                        00C6  6592 ;*
                        00C6  6593 ; The following micro words are used for this SPA Gate Array test.
                        00C6  6594 ;-
                        00C6  6595
                        00C6  6596 1$:
U  0, 7700,C800,0364,0300,0470,1801     00C6  6597 ;*  00  NOP                              ;NOP Micro instruction
U  0, 7700,7800,7777,7777,0470,1802     00D1  6601 ;*  01  LONLIT_[011111111]
U  0, 7700,4800,03C7,03D4,0470,1803     00DC  6605 ;*  02  Q_LONLIT
                        00E7  6609 2$:
U  0, 7700,C840,02C7,0350,0470,1804     00E7  6610 ;*  03  R[14]_Q Q_D                     ;RSRC/12$
U  0, 7700,F800,6EEE,EEEE,8470,1805     00F2  6614 ;*  04  LONLIT_[022222222]
U  0, 7700,4800,03C7,03D4,0470,1806     00FD  6618 ;*  05  Q_LONLIT
                        0108  6622 3$:
U  0, 7700,C840,02C7,0390,0470,1807     0108  6623 ;*  06  R[24]_Q Q_D                     ;RSRC/13$
U  0, 7700,B800,5DDD,DDDD,8470,1808     0113  6627 ;*  07  LONLIT_[044444444]
U  0, 7700,4800,03C7,03D4,0470,1809     011E  6631 ;*  08  Q_LONLIT
                        0129  6635 4$:
```

```
U  0. 7700,0840,02C7,0310,0470,180A  0129  6636 ;%  09  R[4]_Q Q_D                          ;RSRC/14$
U  0. 7700,7800,3BBB,BBBB,8470,180B  0134  6640 ;%  0A  LONLIT_[088888888]
U  0. 7700,4860,0247,03D4,0470,180C  013F  6644 ;%  0B  M[0]_LONLIT
                                     014A  6648 5$:
U  0. 7700,F800,7FFF,FFFD,8470,180D  014A  6649 ;%  0C  LONLIT_[4]                          ;RSRC/15$
U  0. 7700,881D,0047,03D4,0470,180E  0155  6653 ;%  0D  MSRC/RNUM_WBUS,WB_LONLIT
U  0. 7700,0800,0021,0300,0470,180F  0160  6657 ;%  0E  Q_M[0].AND.R[0]
                                     016B  6661 6$:
U  0. 7700,8800,03A5,03C0,0470,1810  016B  6662 ;%  0F  Q_R[30].OR.Q                        ;RSRC/16$
                                     0176  6666 7$:
U  1. 7700,0800,03A5,0350,0470,1811  0176  6667 ;%  10  Q_R[14].OR.Q                        ;RSRC/12$
                                     0181  6671 8$:
U  1. 6700,0800,03A5,0390,0470,1812  0181  6672 ;%  11  RDM_WB,Q_R[24].OR.Q                 ;RSRC/13$
U  1. 7300,C800,0364,0300,0470,1813  018C  6676 ;%  12  NOP,STOP.CLOCK                      ;Stop clock and check results
                                     0197  6680
                               14    0197  6681 12$:     .Byte  ^x14                      ;R[14]  INIT DATA-TABLE 1
                               11    0198  6682           .Byte  ^x11                      ;R[11]
                               19    0199  6683           .Byte  ^x19                      ;R[19]
                               13    019A  6684           .Byte  ^x13                      ;R[13]
                               1C    019B  6685           .Byte  ^x1C                      ;R[1C]
                                     019C  6686
                               24    019C  6687 13$:     .Byte  ^x24                      ;R[24] INIT DATA-TABLE 2
                               21    019D  6688           .Byte  ^x21                      ;R[21]
                               29    019E  6689           .Byte  ^x29                      ;R[29]
                               23    019F  6690           .Byte  ^x23                      ;R[23]
                               2C    01A0  6691           .Byte  ^x2C                      ;R[2C]
                                     01A1  6692
                               04    01A1  6693 14$:     .Byte  ^x4                       ;R[4] INIT DATA-TABLE 3
                               01    01A2  6694           .Byte  ^x1                       ;R[1]
                               09    01A3  6695           .Byte  ^x9                       ;R[9]
                               03    01A4  6696           .Byte  ^x3                       ;R[3]
                               0C    01A5  6697           .Byte  ^xC                       ;R[C]
                                     01A6  6698
                               04    01A6  6699 15$:     .Byte  ^x4                       ; RNUM DATA-TABLE 4
                               01    01A7  6700           .Byte  ^x1
                               08    01A8  6701           .Byte  ^x8
                               03    01A9  6702           .Byte  ^x3
                               01    01AA  6703           .Byte  ^x1
                                     01AB  6704
                               30    01AB  6705 16$:     .Byte  ^x30                      ;TEMP.R RSRC READ DATA-TABLE 5
                               30    01AC  6706           .Byte  ^x30                      ;TEMP.R
                               38    01AD  6707           .Byte  ^x38                      ;TEMP.ROR.1
                               38    01AE  6708           .Byte  ^x38                      ;TEMP.ROR.1
                               2C    01AF  6709           .Byte  ^x2C                      ;R[2C]
                                     01B0  6710
                         FFFFFFFF    01B0  6711 17$:     .Long  ^xFFFFFFFF                ;Results
                         FFFFFFFF    01B4  6712           .Long  ^xFFFFFFFF                ;Results
                         FFFFFFFF    01B8  6713           .Long  ^xFFFFFFFF                ;Results
                         FFFFFFFF    01BC  6714           .Long  ^xFFFFFFFF                ;Results
                         BBBBBBBB    01C0  6715           .Long  ^xBBBBBBBB                ;Results
                                     01C4  6716
                                     01C4  6717 END_TEST_DATA
                                     01C4
                                     01C4       ;-----------------------------------------------------------------------------
                                     01C4  6718
```

```
ZZ-ECKAB-9.2    "TEST 025 SPA Gate Array Test #4                13-MAY-1987       Fiche 1 Frame K14          Sequence 179
ECKAB                      VAX-11/750 DPM MICRO DIAGNOSTICS       11-DEC-1985 09:32:54  VAX/VMS Macro V04-00          Page 15
V09.01                     "TEST 025 SPA Gate Array Test #4       11-DEC-1985 09:32:27  [VAX750.DPM]ECKAB1.MAR;1           (1

                                01C4  6719 ENDTEST
```

```
001B       .SBTTL  "TEST   026     SPA Gate Array Test #5
001B  6721 ;*******************************************************************************
001B  6722 ;++
001B  6723 ;
001B  6724 ; FUNCTIONAL DESCRIPTION
001B  6725 ;
001B  6726 ;       This test checks the SPA Gate Array for proper operation of the chip
001B  6727 ;       selects when indexing by the RNUM Register during a Read M Scratchpad
001B  6728 ;       operation.
001B  6729 ;
001B  6730 ; TEST ALGORITHM
001B  6731 ;
001B  6732 ;       1. Load the micro words to perform the test.
001B  6733 ;       2. Setup a loop of 4 iterations.
001B  6734 ;       3. Select and load the fields into the micro word.
001B  6735 ;       4. Load DCS and execute the micro sequence.
001B  6736 ;               1. Load R[10] Register with 11111111 pattern.
001B  6737 ;               2. Load R[20] Register with 22222222 pattern.
001B  6738 ;               3. Load R[0] with 88888888.
001B  6739 ;               4. Load M Register selected from first table with
001B  6740 ;                  88888888 pattern.
001B  6741 ;               5. Load RNUM Register with value selected from second table.
001B  6742 ;               6. 'AND' M Scratchpad referenced by MSRC field value selected
001B  6743 ;                  from third table with R[0] and  put results in Q
001B  6744 ;                  Register.
001B  6745 ;               7. 'OR' Q Register with R Scratchpad selected from second
001B  6746 ;                  table and put results back in Q Register.
001B  6747 ;               8. 'OR' Q Register with R Scratchpad selected from first
001B  6748 ;                  table and put results in RDM.
001B  6749 ;       5. Check the results.
001B  6750 ;       6. Looped 4 times? If not go to STEP 3.
001B  6751 ;
001B  6752 ; DATA TABLES
001B  6753 ;
001B  6754 ;               M[1]                                    ;MSRC INIT DATA-TABLE 1
001B  6755 ;               M[3]
001B  6756 ;               M[1]
001B  6757 ;               M[1]
001B  6758 ;
001B  6759 ;               8                                       ;RNUM INIT DATA-TABLE 2
001B  6760 ;               2
001B  6761 ;               3
001B  6762 ;               1
001B  6763 ;
001B  6764 ;               M[1]                                    ;TEST DATA-TABLE 3
001B  6765 ;               TEMP.R+1
001B  6766 ;               M[1]
001B  6767 ;               M[1]
001B  6768 ;
001B  6769 ; DATA PATTERNS
001B  6770 ;
001B  6771 ;       The following patterns are used for all tests:
001B  6772 ;
001B  6773 ;               11111111
001B  6774 ;               22222222
```

```
001B  6775 ;                    88888888
001B  6776 ;
001B  6777 ; SIMULATION
001B  6778 ;
001B  6779 ;       This test was developed using output from the Fault Detection
001B  6780 ;       Simulator XLOG with the following parameters:
001B  6781 ;
001B  6782 ;               PHASE=READ
001B  6783 ;               SPW=MLONG
001B  6784 ;               RNUM=8,2,3,1
001B  6785 ;               MSRC=M[1],TEMP.R+1,M[1],M[1]
001B  6786 ;               RSRC=0
001B  6787 ;
001B  6788 ;       Expected chip selects:
001B  6789 ;
001B  6790 ;               RTEMP,MTEMP
001B  6791 ;
001B  6792 ;       Expected MSPA Address from SPA Gate Array:
001B  6793 ;
001B  6794 ;               1
001B  6795 ;               3
001B  6796 ;               1
001B  6797 ;               1
001B  6798 ;
001B  6799 ;       Expected RSPA Address from SPA Gate Array:
001B  6800 ;
001B  6801 ;               0
001B  6802 ;
001B  6803 ; LOGIC DESCRIPTION
001B  6804 ;
001B  6805 ;       This test checks that the SPA Gate Array asserts the proper
001B  6806 ;       chip selects when the RNUM Register and MSRC field is loaded.
001B  6807 ;
001B  6808 ; ASSUMPTIONS
001B  6809 ;
001B  6810 ;       This test requires no signals from any other CPU modules.
001B  6811 ;       It assumes that previous tests have been run to verify that
001B  6812 ;       the logic and the registers used by this test are opeational
001B  6813 ;
001B  6814 ; ERROR DESCRIPTION
001B  6815 ;
001B  6816 ;       Data Expected:BBBBBBBB
001B  6817 ;       Data Received:
001B  6818 ;       Loop Count:
001B  6819 ;
001B  6820 ;--
001B  6821 ;*******************************************************************************
001B  6822 ;///////////////////////////////////////////////////////////////////////////////
001B  6823
001B  6824         INITIALIZE                          ;DCLO Low,Init scratchpads
001D  6825         LOOP    I,1,4                       ;Setup !oop
0024  6826         LDFIELD 5$,I,Byte,2$,64,68          ;Load MSRC micro word field
0035  6827         LDFIELD 6$,I,Byte,3$,31,34,LONLIT       ;Load LONLIT micro word field
0046  6828         LDFIELD 7$,I,Byte,4$,64,68          ;Load MSRC micro word field
0057  6829         LOADDCS 1$,,0,<^x11>                ;Load the test micro instrs
```

```
                        005E  6830          ERRLOOP                         ;Set the error loop
                        0060  6831          FETCH   0                       ;Fetch the first micro word
                        0065  6832          BRSTCLK <^x10>                  ;Clock stop location 10
                        0069  6833          CMPREG  WHREG,8$                ;Check the received data
                        0072  6834          IFERROR ,<<SPA>>                ;Continue if no error else call out SPA
                        0078  6835          ENDLOOP I                       ;Continue with loop
                        007B  6836          SKIP                            ;End of test
                        0082  6837
                        0082  6838 BEGIN_TEST_DATA
                        0082
                        0082          ;-----------------------------------------------------------------------
                        0082  6839 ;+
                        0082  6840 ; The following micro words are used to test the SPA Gate Array.
                        0082  6841 ;-
                        0082  6842
                        0082  6843 1$:
U 00, 7700,C800,0364,0300,0470,1801     0082  6844 ;% 00    NOP                             ;NOP Micro instruction
U 01, 7700,7800,7777,7777,0470,1802     008D  6848 ;% 01    LONLIT_[11111111]
U 02, 7700,4800,03C7,03D4,0470,1803     0098  6852 ;% 02    Q_LONLIT
U 03, 7700,8840,02C7,0340,0470,1804     00A3  6856 ;% 03    R[10]_Q Q_D                     ;Init R[10]
U 04, 7700,F800,6EEE,EEEE,8470,1805     00AE  6860 ;% 04    LONLIT_[22222222]
U 05, 7700,4800,03C7,03D4,0470,1806     00B9  6864 ;% 05    Q_LONLIT
U 06, 7700,8840,02C7,0380,0470,1807     00C4  6868 ;% 06    R[20]_Q Q_D                     ;Init R[20]
U 07, 7700,7800,3BBB,BBBB,8470,1808     00CF  6872 ;% 07    LONLIT_[88888888]
U 08, 7700,4800,03C7,03D4,0470,1809     00DA  6876 ;% 08    Q_LONLIT
U 09, 7700,4840,02C7,0300,0470,180A     00E5  6880 ;% 09    R[0]_Q Q_D                      ;Init R[0]
                        00F0  6884 2$:
U 0A, 7700,C860,0247,03D4,0470,180B     00F0  6885 ;% 0A    M[0]_LONLIT                     ;MSRC/5$
                        00FB  6889 3$:
U 0B, 7700,F800,7FFF,FFFF,8470,180C     00FB  6890 ;% 0B    LONLIT_[0]                      ;LONLIT/6$
U 0C, 7700,C81D,0364,03D4,0470,180D     0106  6894 ;% 0C    MSRC/RNUM_WBUS,RSRC/LONLIT
                        0111  6898 4$:
U 0D, 7700,8800,0021,0300,0470,180E     0111  6899 ;% 0D    Q_M[0].AND.R[0]                 ;MSRC/7$
U 0E, 7700,4800,03A5,0340,0470,180F     011C  6903 ;% 0E    Q_R[10].OR.Q
U 0F, 6700,C800,03A5,0380,0470,1810     0127  6907 ;% 0F    RDM_WB,Q_R[20].OR.Q
U 10, 7300,4800,0364,0300,0470,1811     0132  6911 ;% 10    NOP,STOP.CLOCK                  ;Stop clock and check results.
                        013D  6915
                     01 013D  6916 5$:      .Byte    1                      ;M[1] INIT DATA
                     03 013E  6917          .Byte    3                      ;M[3]
                     01 013F  6918          .Byte    1                      ;M[1]
                     01 0140  6919          .Byte    1                      ;M[1]
                        0141  6920
                     08 0141  6921 6$:      .Byte    8                      ;INIT DATA
                     02 0142  6922          .Byte    2
                     03 0143  6923          .Byte    3
                     01 0144  6924          .Byte    1
                        0145  6925
                     01 0145  6926 7$:      .Byte    1                      ;M[1] INIT DATA
                     3C 0146  6927          .Byte  ^x3C                     ;TEMP.R+1
                     01 0147  6928          .Byte    1                      ;M[1]
                     01 0148  6929          .Byte    1                      ;M[1]
                        0149  6930
               BBBBBBBB 0149  6931 8$:      .Long  ^XBBBBBBBB               ;RESULTS
                        014D  6932
                        014D  6933 END_TEST_DATA
```

```
                        014D
                        014D            ;-----------------------------------------------------------------------------
                        014D    6934
                        014D    6935 ENDTEST
```

```
                        001B        .SBTTL  "TEST   027     SPA Gate Array Test #6
                        001B  6937 ;*******************************************************************************
                        001B  6938 ;++
                        001B  6939 ;
                        001B  6940 ; FUNCTIONAL DESCRIPTION
                        001B  6941 ;
                        001B  6942 ;       This test checks that the SPA Gate Array asserts the correct
                        001B  6943 ;       chip selects during an MTEMP scratchpad write with the RNUM
                        001B  6944 ;       Register set to one of three selected values.
                        001B  6945 ;
                        001B  6946 ; TEST ALGORITHM
                        001B  6947 ;
                        001B  6948 ;       1. Load the micro words to perform the test.
                        001B  6949 ;       2. Setup a loop of 3 iterations.
                        001B  6950 ;       3. Select and load RNUM Register.
                        001B  6951 ;       4. Execute the micro sequence.
                        001B  6952 ;               1. Load R[10] Register with 11111111.
                        001B  6953 ;               2. Load R[20] Register with 22222222.
                        001B  6954 ;               3. Load RNUM Register with selected RNUM Register
                        001B  6955 ;                  value.
                        001B  6956 ;               4. Load M[0] with 88888888 and '3C' in RSRC field.
                        001B  6957 ;               5. 'AND' M[0] Register with R[0] Register and put results in
                        001B  6958 ;                  Q Register.
                        001B  6959 ;               6. 'OR' contents of R[10] Register with the contents of the
                        001B  6960 ;                  Q Register and put the results back in the Q Register.
                        001B  6961 ;               7. 'OR' contents of R[20] Register with the contents of the
                        001B  6962 ;                  Q Register and put the results in the RDM.
                        001B  6963 ;       5. Check the results.
                        001B  6964 ;       6. Looped 3 times? If not go to STEP 3.
                        001B  6965 ;
                        001B  6966 ; DATA PATTERNS
                        001B  6967 ;
                        001B  6968 ;       The following patterns are used:
                        001B  6969 ;
                        001B  6970 ;       11111111
                        001B  6971 ;       22222222
                        001B  6972 ;       88888888
                        001B  6973 ;
                        001B  6974 ; SIMULATION
                        001B  6975 ;
                        001B  6976 ;       This test was developed using output from the Fault Detection
                        001B  6977 ;       Simulator XLOG with the following parameters:
                        001B  6978 ;
                        001B  6979 ;               PHASE=WRITE
                        001B  6980 ;               SPW=MLONG
                        001B  6981 ;               RNUM=1,8,3
                        001B  6982 ;               MSRC=0
                        001B  6983 ;               RSRC=TEMP.R+1
                        001B  6984 ;
                        001B  6985 ;       Expected chip selects:
                        001B  6986 ;
                        001B  6987 ;               RTEMP,MTEMP
                        001B  6988 ;
                        001B  6989 ;       Expected MSPA Address from SPA Gate Array:
                        001B  6990 ;
```

```
                        001B  6991 ;               0
                        001B  6992 ;
                        001B  6993 ;       Expected RSPA Address from SPA Gate Array:
                        001B  6994 ;
                        001B  6995 ;               0
                        001B  6996 ;
                        001B  6997 ; LOGIC DESCRIPTION
                        001B  6998 ;
                        001B  6999 ;       This test checks that the SPA Gate Array asserts only
                        001B  7000 ;       RCS TMP L and MCS TMP L chip selects during an M Scratchpad
                        001B  7001 ;       write with one of three values in the RNUM Register.
                        001B  7002 ;
                        001B  7003 ; ASSUMPTIONS
                        001B  7004 ;
                        001B  7005 ;       This test requires no signals from any other CPU modules.
                        001B  7006 ;       It assumes that previous tests have been run to verify that
                        001B  7007 ;       the logic and registers used by this test are operational.
                        001B  7008 ;
                        001B  7009 ; ERROR DESCRIPTION
                        001B  7010 ;
                        001B  7011 ;       Data Expected:BBBBBBBB
                        001B  7012 ;       Data Received:
                        001B  7013 ;       Loop Count:
                        001B  7014 ;
                        001B  7015 ;--
                        001B  7016 ;*******************************************************************************
                        001B  7017 ;///////////////////////////////////////////////////////////////////////////////
                        001B  7018
                        001B  7019         INITIALIZE                          ;DCLO Low,Init scratchpads
                        001D  7020         LOOP    I,1,3                       ;Setup loop
                        0024  7021         LDFIELD 3$,I,Byte,2$,31,34,LONLIT       ;Load micro word field
                        0035  7022         LOADDCS 1$,,0,<^xF>                 ;Load the test micro instrs
                        003C  7023         ERRLOOP                             ;Set the error loop
                        003E  7024         FETCH   0                           ;Fetch the first micro word
                        0043  7025         BRSTCLK <^xE>                       ;Clock stop location E
                        0047  7026         CMPREG  WHREG,4$,,Long              ;Check the received data
                        0050  7027         IFERROR ,<<SPA>>                    ;Continue if no error else call out SPA
                        0056  7028         ENDLOOP I                           ;Continue with loop
                        0059  7029         SKIP                                ;End of test
                        0060  7030
                        0060  7031 BEGIN_TEST_DATA
                        0060
                        0060         ;-------------------------------------------------------------------------
                        0060  7032 ;+
                        0060  7033 ; The following micro words are used to test the SPA Gate Array.
                        0060  7034 ;-
                        0060  7035
                        0060  7036 1$:
U 00, 7700,C800,0364,0300,0470,1801     0060  7037 ;* 00    NOP                             ;NOP Micro instruction
U 01, 7700,7800,7777,7777,0470,1802     006B  7041 ;* 01    LONLIT_[11111111]
U 02, 7700,4800,03C7,03D4,0470,1803     0076  7045 ;* 02    Q_LONLIT
U 03, 7700,8840,02C7,0340,0470,1804     0081  7049 ;* 03    R[10]_Q Q_D                     ;Load R[10]
U 04, 7700,F800,6EEE,EEEE,8470,1805     008C  7053 ;* 04    LONLIT_[22222222]
U 05, 7700,4800,03C7,03D4,0470,1806     0097  7057 ;* 05    Q_LONLIT
U 06, 7700,8840,02C7,0380,0470,1807     00A2  7061 ;* 06    R[20]_Q Q_D                     ;Load R[20]
```

```
U 07, 7700,7800,3BBB,BBBB,8470,1808  00AD  7065 ;% 07   LONLIT_[88888888]
U 08, 7700,4860,0247,03D4,0470,1809  00B8  7069 ;% 08   M[0]_LONLIT                    ;Load M[0]
                                     00C3  7073 2$:
U 09, 7700,F800,7FFF,FFFF,8470,180A  00C3  7074 ;% 09   LONLIT_[0]                    ;LONLIT/a1
U 0A, 7700,C81D,0057,03D4,0470,180B  00CE  7078 ;% 0A   RNUM_LONLIT
U 0B, 7700,0800,0021,0300,0470,180C  00D9  7082 ;% 0B   Q_M[0].AND.R[0]
U 0C, 7700,C800,03A5,0340,0470,180D  00E4  7086 ;% 0C   Q_R[10].OR.Q
U 0D, 6700,C800,03A5,0380,0470,180E  00EF  7090 ;% 0D   RDM_WB,Q_R[20].OR.Q
U 0E, 7300,4800,0364,0300,0470,180F  00FA  7094 ;% 0E   NOP,STOP.CLOCK                ;Stop clock and check results
                                     0105  7098
                                     0105  7099 ;RNUM Register select values
                                     0105  7100
                                  01 0105  7101 3$:     .Byte  ^x1
                                  02 0106  7102         .Byte  ^x2
                                  08 0107  7103         .Byte  ^x8
                                     0108  7104
                            BBBBBBBB 0108  7105 4$:     .Long  ^xBBBBBBBB            ;Results
                                     010C  7106
                                     010C  7107 END_TEST_DATA
                                     010C
                                     010C       ;-----------------------------------------------------------------------
                                     010C  7108
                                     010C  7109 ENDTEST
```

```
                        0017        .SBTTL  "TEST   028     ALU LOOP Flag Test
                        0017  7111 ;*******************************************************************************
                        0017  7112 ;++
                        0017  7113 ;
                        0017  7114 ; FUNCTIONAL DESCRIPTION
                        0017  7115 ;
                        0017  7116 ;       This test checks that the LOOP Flag in the ALK Gate Array can be set
                        0017  7117 ;       and cleared.
                        0017  7118 ;
                        0017  7119 ; TEST ALGORITHM
                        0017  7120 ;
                        0017  7121 ;       1. Load the micro words to perform the test.
                        0017  7122 ;       2. Setup a loop of three iterations.
                        0017  7123 ;       3. Select and load ALPCTL Field with data from Table 1.
                        0017  7124 ;       4. Execute the micro sequence.
                        0017  7125 ;               1. Set the ALU LOOP Flag.
                        0017  7126 ;               2. Read LOOP Flag onto WBUS.
                        0017  7127 ;       5. Check the results.
                        0017  7128 ;
                        0017  7129 ; DATA PATTERNS for ALPCTL Field
                        0017  7130 ;
                        0017  7131 ;                       279     MUL
                        0017  7132 ;                       26C     DIV
                        0017  7133 ;                       26F     DIVD
                        0017  7134 ;
                        0017  7135 ; LOGIC DESCRIPTION
                        0017  7136 ;
                        0017  7137 ;       This test checks the ALK Gate Array's LOOP Flag circuitry.
                        0017  7138 ;
                        0017  7139 ; ASSUMPTIONS
                        0017  7140 ;
                        0017  7141 ;       This test requires no signals from any other CPU modules but assumes an
                        0017  7142 ;       operational RBUS, MBUS and WBUS.
                        0017  7143 ;
                        0017  7144 ; ERROR DESCRIPTION
                        0017  7145 ;
                        0017  7146 ;       Data expected:          Loop 1          Flag set
                        0017  7147 ;                               Loop 2          Flag set
                        0017  7148 ;                               Loop 3          Flag clear
                        0017  7149 ;       Data received:
                        0017  7150 ;--
                        0017  7151 ;*******************************************************************************
                        0017  7152 ;///////////////////////////////////////////////////////////////////////////////
                        0017  7153
                        0017  7154         INITIALIZE                              ;DCLO low,Init scratchpads
                        0019  7155         LOOP    I,1,3                           ;Set loop count
                        0020  7156         LDFIELD 3$,I,Word,2$,48,57              ;Load Field
                        0031  7157         LOADDCS 1$,,0,4                         ;Load the test micro instrs.
                        0038  7158         ERRLOOP                                 ;Set the error loop
                        003A  7159         FETCH   0                               ;Get the first micro word
                        003F  7160         BRSTCLK 3                               ;Clock stop location 3
                        0043  7161         CMPREGMSK WHREG,4$,,5$,,Long            ;Check the received data
                        004F  7162         IFERROR ,<<ALK>>                        ;Continue if no error else call out ALK
                        0055  7163         ENDLOOP I                               ;Continue with loop
                        0058  7164         SKIP                                    ;End of test
```

```
                                005F  7165
                                005F  7166 BEGIN_TEST_DATA
                                005F
                                005F        ;-------------------------------------------------------------------------
                                005F  7167 ;*
                                005F  7168 ;The following micro words are used to test the LOOP Flag located within the
                                005F  7169 ;ALK Gate Array.
                                005F  7170 ;-
                                005F  7171
                                005F  7172 1$:
U 00, 7700,C800,0364,0300,0470,1801  005F  7173 ;* 00    NOP
                                006A  7177 2$:
U 01, 7700,C800,0364,0300,0470,1802  006A  7178 ;* 01    ALPCTL/NOP                      ; Load ALPCTL Field
U 02, 6700,0800,0378,0300,0470,1803  0075  7182 ;* 02    ALPCTL/WB_LOOPF,RDM_WB          ; Is LOOP Flag set?
U 03, 7300,C800,0364,0300,0470,1804  0080  7186 ;* 03    NOP,STOP.CLOCK                  ; Stop clock and check results
                                008B  7190
                            40000000  008B  7191 3$:      .Long   ^x40000000              ; Bit 30 is set
                            40000000  008F  7192          .Long   ^x40000000              ;  ditto
                            00000000  0093  7193          .Long             0             ; Bit 30 is cleared
                                0097  7194
                                0279  0097  7195 4$:      .Word   ^x279                   ; DIV  -TABLE 1-
                                026C  0099  7196          .Word   ^x26C                   ; MUL
                                026F  009B  7197          .Word   ^x26F                   ; DIVD
                                009D  7198
                            40000000  009D  7199 5$:      .Long   ^x40000000              ; Look only at bit 30
                                00A1  7200
                                00A1  7201 END_TEST_DATA
                                00A1
                                00A1        ;-------------------------------------------------------------------------
                                00A1  7202
                                00A1  7203 ENDTEST
```

```
                        0016        .SBTTL  "TEST   029     Test PSL Register
                        0016  7205 ;*******************************************************************************
                        0016  7206 ;++
                        0016  7207 ;
                        0016  7208 ; FUNCTIONAL DESCRIPTION:
                        0016  7209 ;
                        0016  7210 ;       This test will check the integrity of the Processor Status Longword
                        0016  7211 ;       Register by writing and reading 6 test patterns through the Register.
                        0016  7212 ;
                        0016  7213 ;
                        0016  7214 ; TEST ALGORITHM:
                        0016  7215 ;
                        0016  7216 ;       1.  CREATE A TEST LOOP OF 6 ITERATIONS
                        0016  7217 ;       2.  LOAD A TEST PATTERN INTO THE WD REGISTER (RDM)
                        0016  7218 ;       3.  EXECUTE DCS PROGRAM
                        0016  7219 ;               1.  WRITE TEST PATTERN IN WD REGISTER (RDM) INTO PSL REGISTER
                        0016  7220 ;               2.  READ TEST PATTERN BACK TO RDM
                        0016  7221 ;       4.  TEST VBUS FOR PSL CM H IN CORRECT STATE
                        0016  7222 ;       5.  IF NO ERROR COMPARE PSL DATA
                        0016  7223 ;       6.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
                        0016  7224 ;
                        0016  7225 ;
                        0016  7226 ; TEST PATTERNS:
                        0016  7227 ;
                        0016  7228 ;       ITERATION       TEST PATTERN
                        0016  7229 ;           1             AAAAAAAA
                        0016  7230 ;           2             55555555
                        0016  7231 ;           3             33333333
                        0016  7232 ;           4             0F0F0F0F
                        0016  7233 ;           5             00FF00FF
                        0016  7234 ;           6             0000FFFF
                        0016  7235 ;
                        0016  7236 ;
                        0016  7237 ; LOGIC DESCRIPTION:
                        0016  7238 ;
                        0016  7239 ;       THIS TEST CHECKS THE PROCESSOR STATUS LONGWORD REGISTER BY USING
                        0016  7240 ;       THE "CCPSL" FIELD TO WRITE AND READ TEST PATTERNS TO THE REGISTER.
                        0016  7241 ;       IF FAILURES OCCUR YOU CAN USE THE FOLLOWING CHART TO DETERMINE
                        0016  7242 ;       WHICH GATE ARRAY FAILED:
                        0016  7243 ;
                        0016  7244 ;               BITS            GATE ARRAY
                        0016  7245 ;               3:0               CCC
                        0016  7246 ;               4                 PHB
                        0016  7247 ;               7:5               CCC
                        0016  7248 ;               8:15              MBZ *
                        0016  7249 ;               16:20             INT
                        0016  7250 ;               21                MBZ *
                        0016  7251 ;               26:22             INT
                        0016  7252 ;               27                PHB
                        0016  7253 ;               28:29             MBZ *
                        0016  7254 ;               31:30             PHB
                        0016  7255 ;
                        0016  7256 ;       *  MBZ = MUST BE ZERO
                        0016  7257 ;
                        0016  7258 ;
```

```
                     0016  7259 ; ASSUMPTIONS:
                     0016  7260 ;
                     0016  7261 ;       THIS TEST ASSUMES THE WBUS IS OPERATIONAL.
                     0016  7262 ;
                     0016  7263 ;
                     0016  7264 ; ERROR DESCRIPTION:
                     0016  7265 ;
                     0016  7266 ;       FIRST IFERROR:
                     0016  7267 ;               EXPECTED VBUS DATA (PSL CM H)
                     0016  7268 ;               RECEIVED VBUS DATA
                     0016  7269 ;               LOOP COUNT I
                     0016  7270 ;                       (BITS <7:0> = BIT POSITION)
                     0016  7271 ;                       (BITS <8> = BIT STATE)
                     0016  7272 ;
                     0016  7273 ;       SECOND IFERROR:
                     0016  7274 ;               EXPECTED PSL DATA
                     0016  7275 ;               RECEIVED PSL DATA
                     0016  7276 ;               LOOP COUNT I
                     0016  7277 ;
                     0016  7278 ;--
                     0016  7279 ;*******************************************************************************
                     0016  7280 ;///////////////////////////////////////////////////////////////////////////////
                     0016  7281
                     0016  7282         INITIALIZE                                      ;INIT CPU
                     0018  7283         LOOP            I,1,6                           ;CREATE A TEST LOOP
                     001F  7284         SA01PAT         I,1$,,                          ;GENERATE A TEST PATTERN
                     0027  7285         LOADREG         WDREG,1$,,L                     ;LOAD TEST PATTERN INTO WD REG
                     002F  7286         ERRLOOP                                         ;ERROR LOOP
                     0031  7287         FETCH           0                               ;START DCS PROGRAM
                     0036  7288         BRSTCLK         3                               ;END DCS PROGRAM
                     003A  7289         CMPVBUS         3$,I                            ;TEST PSL CM SIGNAL
                     003F  7290         IFERROR         ,<PHB>                          ;SIGNAL NOT PRESENT
                     0045  7291         CMPREGMSK       WHREG,1$,,2$,,L,                ;COMPARE RESULTS
                     0051  7292         IFERROR         ,<<CCC><INT><PHB>>,<UBI>
                     005A  7293         ENDLOOP         I                               ;END TEST LOOP
                     005D  7294         SKIP                                            ;END OF TEST
                     0064  7295
                     0064  7296 BEGIN_TEST_DATA
                     0064
                     0064       ;-------------------------------------------------------------------------
                     0064  7297
00000000             0064  7298 1$:     .LONG   ^X00000000                              ;TEST PATTERN STORAGE
CFDF00FF             0068  7299 2$:     .LONG   ^XCFDF00FF                              ;MASK FOR CMPREGMSK PSEUDO-OP
       01            006C  7300 3$:     .BYTE   ^X1                                     ;PSL CM SIGNAL
                     006D  7301         VBUSDATA        <^X1F>,1
       01            006E  7302         .BYTE   ^X1
                     006F  7303         VBUSDATA        <^X1F>,0
       01            0070  7304         .BYTE   ^X1
                     0071  7305         VBUSDATA        <^X1F>,0
       01            0072  7306         .BYTE   ^X1
                     0073  7307         VBUSDATA        <^X1F>,0
       01            0074  7308         .BYTE   ^X1
                     0075  7309         VBUSDATA        <^X1F>,0
       01            0076  7310         .BYTE   ^X1
                     0077  7311         VBUSDATA        <^X1F>,0
```

```
                                0078  7312
                                0078  7313 BEGIN_CPU_DATA
                                0002  7314
                                0002  7315 DCS_DATA
                                0001  7316
U 00, 7700,C800,0364,0300,0470,1801  0001  7317 ;% 00   NOP
U 01, 5700,4800,0364,0300,0070,1802  000C  7321 ;% 01   WB_RDM,CCPSL/PSL_WB.CCBR_ALUS
U 02, 6600,4800,0364,0300,0870,1803  0017  7325 ;% 02   RDM_WB,WB_PSL,STROBE.V.BUS
U 03, 7300,C800,0364,0300,0470,1804  0022  7329 ;% 03   NOP,STOP.CLOCK
                                002D  7333
                                002D  7334 END_CPU_DATA
                                0078  7335
                                0078  7336 END_TEST_DATA
                                0078
                                0078       ;-------------------------------------------------------------------------
                                0078  7337
                                0078  7338 ENDTEST
```

```
                        0042        .SBTTL  "TEST   02A     Test MISC, CCMISC, AND CCPSL Micro-Orders That Affect the PS
                        0042  7340 ;*******************************************************************************
                        0042  7341 ;++
                        0042  7342 ;
                        0042  7343 ; FUNCTIONAL DESCRIPTION:
                        0042  7344 ;
                        0042  7345 ;       This test will consist of a series of subtests to verify the
                        0042  7346 ;       correct operation of Micro-orders that directly access
                        0042  7347 ;       any of the PSL Fields.  The Micro-orders used are:
                        0042  7348 ;               1.  MISC:
                        0042  7349 ;                       CLR.TP
                        0042  7350 ;                       CLR.FPD
                        0042  7351 ;                       SET.FPD
                        0042  7352 ;
                        0042  7353 ;               2.  CCMISC:
                        0042  7354 ;                       SETV.CCBR_SIGND
                        0042  7355 ;
                        0042  7356 ;               3.  CCPSL:
                        0042  7357 ;                       CC_WB.CCBR_ALUS
                        0042  7358 ;                       PSW_WB.CCBR_ALUS
                        0042  7359 ;
                        0042  7360 ;
                        0042  7361 ; TEST ALGORITHM:
                        0042  7362 ;
                        0042  7363 ;       1.  MODIFY DCS FOR THIS SUBTEST
                        0042  7364 ;       2.  LOAD WD REGISTER WITH INITIAL DATA PATTERN FOR PSL REGISTER
                        0042  7365 ;       3.  EXECUTE DCS PROGRAM
                        0042  7366 ;               1.  WRITE DATA PATTERN IN WD REGISTER (RDM) INTO PSL
                        0042  7367 ;               2.  PERFORM TEST ON PSL REGISTER
                        0042  7368 ;               3.  READ PSL REGISTER BACK TO RDM
                        0042  7369 ;       4.  COMPARE RESULTS (EXPECTED AND RECEIVED)
                        0042  7370 ;       5.  IF NO ERROR GO TO NEXT SUBTEST
                        0042  7371 ;
                        0042  7372 ;
                        0042  7373 ; TEST PATTERNS:
                        0042  7374 ;
                        0042  7375 ;       SUBTEST         MICRO-ORDER             PSL BITS AFFECTED
                        0042  7376 ;
                        0042  7377 ;          1            MISC/CLR.TP                     30
                        0042  7378 ;          2            MISC/CLR.FPD                    27
                        0042  7379 ;          3            MISC/SET.FPD                    27
                        0042  7380 ;          4            CCMISC/SETV.CCBR_SIGND          01
                        0042  7381 ;          5            CCPSL/CC_WB.CCBR_ALUS           03:00
                        0042  7382 ;          6            CCPSL/PSW_WB.CCBR_ALUS          15:00
                        0042  7383 ;
                        0042  7384 ;
                        0042  7385 ; ASSUMPTIONS:
                        0042  7386 ;
                        0042  7387 ;       THIS TEST ASSUMES THE WBUS IS OPERATIONAL.
                        0042  7388 ;
                        0042  7389 ;
                        0042  7390 ; ERROR DESCRIPTION:
                        0042  7391 ;
                        0042  7392 ;       EXPECTED PSL DATA
                        0042  7393 ;       RECEIVED PSL DATA
```

```
                     7394 ;
                     7395 ;--
                     7396 ;*****************************************************************************
                     7397
                     7398         SUBTEST 1
                     7399 ;+
                     7400 ;SUBTEST #1
                     7401 ;
                     7402 ;       TEST "MISC/CLR.TP" MICRO-ORDER
                     7403 ;-
0045  7404           INITIALIZE                                      ;INIT CPU
0047  7405           LOADDCS         14$,,02,1                       ;MODIFY DCS ADDRESS 02
004E  7406           LOADREG         WDREG,1$,,L                     ;PSL DATA TO WD REGISTER
0056  7407           ERRLOOP                                         ;ERROR LOOP
0058  7408           FETCH           0                               ;START DCS PROGRAM
005D  7409           BRSTCLK         4                               ;END DCS PROGRAM
0061  7410           CMPREGMSK       WHREG,2$,,15$,,L,               ;COMPARE RESULTS
006D  7411           IFERROR         ,<<PHB>>                        ;FAILING ARRAY AND MODULE
0073  7412
                     7413         SUBTEST 2
                     7414 ;+
                     7415 ;SUBTEST #2
                     7416 ;
                     7417 ;       TEST "MISC/CLR.FPD" MICRO-ORDER
                     7418 ;-
0076  7419           INITIALIZE                                      ;INIT CPU
0078  7420           LOADDCS         3$,,02,1                        ;MODIFY DCS ADDRESS 02
007F  7421           ERRLOOP                                         ;ERROR LOOP
0081  7422           FETCH           0                               ;START DCS PROGRAM
0086  7423           BRSTCLK         4                               ;END DCS PROGRAM
008A  7424           CMPREGMSK       WHREG,4$,,15$,,L,               ;COMPARE RESULTS
0096  7425           IFERROR         ,<<PHB>>                        ;FAILING ARRAY AND MODULE
009C  7426
                     7427         SUBTEST 3
                     7428 ;+
                     7429 ;SUBTEST #3
                     7430 ;
                     7431 ;       TEST "MISC/SET.FPD" MICRO-ORDER
                     7432 ;-
009F  7433           INITIALIZE                                      ;INIT CPU
00A1  7434           LOADREG         WDREG,5$,,L                     ;PSL DATA TO WD REGISTER
00A9  7435           LOADDCS         6$,,02,1                        ;MODIFY DCS ADDRESS 02
00B0  7436           ERRLOOP                                         ;ERROR LOOP
00B2  7437           FETCH           0                               ;START DCS PROGRAM
00B7  7438           BRSTCLK         4                               ;END DCS PROGRAM
00BB  7439           CMPREGMSK       WHREG,7$,,15$,,L,               ;COMPARE RESULTS
00C7  7440           IFERROR         ,<<CCC>>                        ;FAILING ARRAY AND MODULE
00CD  7441
                     7442         SUBTEST 4
```

```
                        00D0
                        00D0        ;//////////////////////////////////////////////////////////////////////////////
                        00D0  7443  ;+
                        00D0  7444  ;SUBTEST #4
                        00D0  7445  ;
                        00D0  7446  ;       TEST "CCMISC/SETV.CCBR_SIGND" MICRO-ORDER
                        00D0  7447  ;-
                        00D0  7448          INITIALIZE                                  ;INIT CPU
                        00D2  7449          LOADDCS         8$,,02,1                    ;MODIFY DCS ADDRESS 02
                        00D9  7450          ERRLOOP                                     ;ERROR LOOP
                        00DB  7451          FETCH           0                           ;START DCS PROGRAM
                        00E0  7452          BRSTCLK         4                           ;END DCS PROGRAM
                        00E4  7453          CMPREGMSK       WHREG,9$,,15$,,L,           ;COMPARE RESULTS
                        00F0  7454          IFERROR         ,<<CCC>>                    ;FAILING ARRAY AND MODULE
                        00F6  7455
                        00F6  7456          SUBTEST 5
                        00F9
                        00F9        ;//////////////////////////////////////////////////////////////////////////////
                        00F9  7457  ;+
                        00F9  7458  ;SUBTEST #5
                        00F9  7459  ;
                        00F9  7460  ;       TEST "CCPSL/CC_WB.CCBR_ALUS" MICRO-ORDER
                        00F9  7461  ;-
                        00F9  7462          INITIALIZE                                  ;INIT CPU
                        00FB  7463          LOADDCS         10$,,02,1                   ;MODIFY DCS ADDRESS 02
                        0102  7464          ERRLOOP                                     ;ERROR LOOP
                        0104  7465          FETCH           0                           ;START DCS PROGRAM
                        0109  7466          BRSTCLK         4                           ;END DCS PROGRAM
                        010D  7467          CMPREGMSK       WHREG,11$,,15$,,L,          ;COMPARE RESULTS
                        0119  7468          IFERROR         ,<<CCC>>                    ;FAILING ARRAY AND MODULE
                        011F  7469
                        011F  7470          SUBTEST 6
                        0122
                        0122        ;//////////////////////////////////////////////////////////////////////////////
                        0122  7471  ;+
                        0122  7472  ;SUBTEST #6
                        0122  7473  ;
                        0122  7474  ;       TEST "CCPSL/PSW_WB.CCBR_ALUS" MICRO-ORDER
                        0122  7475  ;-
                        0122  7476          INITIALIZE                                  ;INIT CPU
                        0124  7477          LOADDCS         12$,,02,1                   ;MODIFY DCS ADDRESS 02
                        012B  7478          ERRLOOP                                     ;ERROR LOOP
                        012D  7479          FETCH           0                           ;START DCS PROGRAM
                        0132  7480          BRSTCLK         4                           ;END DCS PROGRAM
                        0136  7481          CMPREGMSK       WHREG,13$,,15$,,L,          ;COMPARE RESULTS
                        0142  7482          IFERROR         ,<<CCC><PHB>>               ;FAILING ARRAY AND MODULE
                        0149  7483          SKIP                                        ;END OF TEST
                        0150  7484
                        0150  7485  BEGIN_TEST_DATA
                        0150
                        0150        ;------------------------------------------------------------------------------
                        0150  7486
FFFFFFFF                0150  7487  1$:     .LONG   ^XFFFFFFFF                          ;PSL DATA
BFFFFFFF                0154  7488  2$:     .LONG   ^XBFFFFFFF                          ;RESULTS SUBTEST #1
                        0158  7489
```

```
                                0158  7490 3$:
U 02, 7700,4E00,0364,0300,0470,1803  0158  7491 ;% 02   MISC/CLR.FPD                                ;DCS ADD 02 SUBTEST #2
                                0163  7495
                    F7FFFFFF    0163  7496 4$:     .LONG   ^XF7FFFFFF                              ;RESULTS SUBTEST #2
                    00000000    0167  7497 5$:     .LONG   ^X00000000                              ;PSL DATA
                                016B  7498
                                016B  7499 6$:
U 02, 7700,CE80,0364,0300,0470,1803  016B  7500 ;% 02   MISC/SET.FPD                                ;DCS ADD 02 SUBTEST #3
                                0176  7504
                    08000000    0176  7505 7$:     .LONG   ^X08000000                              ;RESULTS SUBTEST #3
                                017A  7506
                                017A  7507 8$:
U 02, 7700,0800,0364,0300,8E70,1803  017A  7508 ;% 02   CCMISC/SETV.CCBR_SIGND                      ;DCS ADD 02 SUBTEST #4
                                0185  7512
                    00000002    0185  7513 9$:     .LONG   ^X00000002                              ;RESULTS SUBTEST #4
                                0189  7514
                                0189  7515 10$:
U 02, 7700,C800,5BE4,0300,0A70,1803  0189  7516 ;% 02   WB_R[TEMP0],CCPSL/CC_WB.CCBR_ALUS           ;DCS ADD 02 SUBTEST #5
                                0194  7520
                    0000000F    0194  7521 11$:    .LONG   ^X0000000F                              ;RESULTS SUBTEST #5
                                0198  7522
                                0198  7523 12$:
U 02, 7700,0800,5BE4,0304,0270,1803  0198  7524 ;% 02   WB_R[TEMP1],CCPSL/PSW_WB.CCBR_ALUS          ;DCS ADD 02 SUBTEST #6
                                01A3  7528
                    0000FFFF    01A3  7529 13$:    .LONG   ^X0000FFFF                              ;RESULTS SUBTEST #6
                                01A7  7530
                                01A7  7531 14$:
U 02, 7700,C900,0364,0300,0470,1803  01A7  7532 ;% 02   MISC/CLR.TP                                 ;DCS ADD 02 SUBTEST #1
                                01B2  7536
                    CFDF00FF    01B2  7537 15$:    .LONG   ^XCFDF00FF                              ;MASK FOR CMPREGMSK PSEUDO-OP
                                01B6  7538
                                01B6  7539 BEGIN_CPU_DATA
                                0002  7540
                                0002  7541 DCS_DATA
                                0001  7542
U 00, 7700,C800,0364,0300,0470,1801  0001  7543 ;% 00   NOP
U 01, 5700,4800,0364,0300,0070,1802  000C  7547 ;% 01   WB_RDM,CCPSL/PSL_WB.CCBR_ALUS              ;WRITE PSL DATA
U 02, 7700,C900,0364,0300,0470,1803  0017  7551 ;% 02   MISC/CLR.TP                                ;MICRO-ORDER BEING TESTED
U 03, 6700,C800,0364,0300,0870,1804  0022  7555 ;% 03   RDM_WB,WB_PSL                              ;READ PSL TO RDM
U 04, 7300,4800,0364,0300,0470,1805  002D  7559 ;% 04   NOP,STOP.CLOCK                             ;STOP CPU CLOCK
                                0038  7563
                                0038  7564 RTEMP_DATA
                                0001  7565
                    0000000F    0001  7566         .LONG   ^X0000000F                              ;TEST PATTERN SUBTEST #5
                    0000FFFF    0005  7567         .LONG   ^X0000FFFF                              ;TEST PATTERN SUBTEST #6
                                0009  7568
                                0009  7569 END_CPU_DATA
                                01B6  7570
                                01B6  7571 END_TEST_DATA
                                01B6
                                01B6       ;-----------------------------------------------------------------------------
                                01B6  7572
                                01B6  7573 ENDTEST
```

```
0039          .SBTTL  "TEST   02B     Test WCTRL Micro-Orders that affect the PSL Register
0039   7575 ;*****************************************************************************
0039   7576 ;++
0039   7577 ;
0039   7578 ; FUNCTIONAL DESCRIPTION:
0039   7579 ;
0039   7580 ;       This test will consist of 3 subtests to check the operation of
0039   7581 ;       the WCTRL Micro-orders that affect the Processor Status Longword
0039   7582 ;       Register.  Micro-orders tested are:
0039   7583 ;                1.  IPL_WB
0039   7584 ;                2.  PREV_WB
0039   7585 ;                3.  PREV_CUR.ISCUR_WB
0039   7586 ;
0039   7587 ;
0039   7588 ; TEST ALGORITHM:
0039   7589 ;
0039   7590 ;       1.  MODIFY DCS FOR THIS SUBTEST
0039   7591 ;       2.  LOAD DATA PATTERN FOR PSL INTO WD REGISTER (RDM)
0039   7592 ;       3.  EXECUTE DCS PROGRAM
0039   7593 ;               1.  WRITE DATA PATTERN IN WD REGISTER TO PSL REGISTER
0039   7594 ;               2.  PERFORM MICRO-ORDER BEING TESTED
0039   7595 ;               3.  READ PSL BACK TO RDM
0039   7596 ;       4.  COMPARE RESULTS (EXPECTED AND RECEIVED)
0039   7597 ;       5.  IF NO ERROR GO TO NEXT SUBTEST
0039   7598 ;
0039   7599 ;
0039   7600 ; TEST PATTERNS:
0039   7601 ;
0039   7602 ;       SUBTEST         MICRO_ORDER             PSL BITS AFFECTED
0039   7603 ;
0039   7604 ;          1            IPL_WB                      20:16
0039   7605 ;          2            PREV_WB                     23:22
0039   7606 ;          3            PREV_CUR.ISCUR_WB           26:22
0039   7607 ;
0039   7608 ;
0039   7609 ; LOGIC DESCRIPTION:
0039   7610 ;
0039   7611 ;       This test exercises only those Processor Status Longword bits which
0039   7612 ;       reside in the "INT" Gate Array.  Previous tests of the PSL Register
0039   7613 ;       have almost totally checked out the related logic.  Any failures in
0039   7614 ;       this test would have to be due to the INT Decode logic. (I hope!)
0039   7615 ;
0039   7616 ;
0039   7617 ; ASSUMPTIONS:
0039   7618 ;
0039   7619 ;       THIS TEST ASSUMES THE WBUS AND DPM ARE OPERATIONAL
0039   7620 ;
0039   7621 ;
0039   7622 ; ERROR DESCRIPTION:
0039   7623 ;
0039   7624 ;       EXPECTED PSL DATA
0039   7625 ;       RECEIVED PSL DATA
0039   7626 ;
0039   7627 ;--
0039   7628 ;*****************************************************************************
```

```
                     0039  7629
                     0039  7630          SUBTEST 1
                     003C
                     003C          ;////////////////////////////////////////////////////////////////////////////
                     003C  7631    ;+
                     003C  7632    ;SUBTEST #1
                     003C  7633    ;
                     003C  7634    ;       TEST "WCTRL/IPL_WB" MICRO-ORDER
                     003C  7635    ;-
                     003C  7636          INITIALIZE                                  ;INIT CPU
                     003E  7637          LOADDCS         8$,,02,1                    ;MODIFY DCS ADDRESS 02
                     0045  7638          LOADREG         WDREG,1$,,L                 ;PSL DATA PATTERN TO WD REG
                     004D  7639          ERRLOOP                                     ;ERROR LOOP
                     004F  7640          FETCH           0                           ;START DCS PROGRAM
                     0054  7641          BRSTCLK         4                           ;END DCS PROGRAM
                     0058  7642          CMPREGMSK       WHREG,2$,,9$,,L,            ;COMPARE RESULTS
                     0064  7643          IFERROR         ,<<INT>>,<<UBI>>            ;FAILING ARRAY AND MODULE
                     006B  7644
                     006B  7645          SUBTEST 2
                     006E
                     006E          ;////////////////////////////////////////////////////////////////////////////
                     006E  7646    ;+
                     006E  7647    ;SUBTEST #2
                     006E  7648    ;
                     006E  7649    ;       TEST "WCTRL/PREV_WB" MICRO-ORDER
                     006E  7650    ;-
                     006E  7651          INITIALIZE                                  ;INIT CPU
                     0070  7652          LOADDCS         3$,,02,1                    ;MODIFY DCS ADDRESS 02
                     0077  7653          ERRLOOP                                     ;ERROR LOOP
                     0079  7654          FETCH           0                           ;START DCS PROGRAM
                     007E  7655          BRSTCLK         4                           ;END DCS PROGRAM
                     0082  7656          CMPREGMSK       WHREG,4$,,9$,,L,            ;COMPARE RESULTS
                     008E  7657          IFERROR         ,<<INT>>,<<UBI>>            ;FAILING ARRAY AND MODULE
                     0095  7658
                     0095  7659          SUBTEST 3
                     0098
                     0098          ;////////////////////////////////////////////////////////////////////////////
                     0098  7660    ;+
                     0098  7661    ;SUBTEST #3
                     0098  7662    ;
                     0098  7663    ;       TEST "WCTRL/PREV_CUR.ISCUR_WB" MICRO-ORDER
                     0098  7664    ;-
                     0098  7665          INITIALIZE                                  ;INIT CPU
                     009A  7666          LOADDCS         5$,,02,1                    ;MIDIFY DCS ADDRESS 02
                     00A1  7667          LOADREG         WDREG,6$,,L                 ;DATA PATTERN FOR PSL FROM
                     00A9  7668                                                      ;WD REGISTER
                     00A9  7669          ERRLOOP                                     ;ERROR LOOP
                     00AB  7670          FETCH           0                           ;START DCS PROGRAM
                     00B0  7671          BRSTCLK         4                           ;END DCS PROGRAM
                     00B4  7672          CMPREGMSK       WHREG,7$,,9$,,L,            ;COMPARE RESULTS
                     00C0  7673          IFERROR         ,<<INT>>,<<UBI>>            ;FAILING ARRAY AND MODULE
                     00C7  7674          SKIP                                        ;END OF TEST
                     00CE  7675
                     00CE  7676    BEGIN_TEST_DATA
                     00CE
```

```
                                  00CE        ;-----------------------------------------------------------------------
                                  00CE
                        00000000  00CE  1$:      .LONG   ^X00000000                        ;PSL DATA
                        001F0000  00D2  2$:      .LONG   ^X001F0000                        ;RESULTS SUBTEST #1
                                  00D6
                                  00D6  3$:
U 02, 7700,0800,5BE4,0304,6270,1803  00D6  ;% 02   WB_R[TEMP1],WCTRL/PREV_WB                 ;DCS ADD 02 SUBTEST #2
                                  00E1
                        00C00000  00E1  4$:      .LONG   ^X00C00000                        ;RESULTS SUBTEST #2
                                  00E5
                                  00E5  5$:
U 02, 7700,8800,5BE4,0308,6A70,1803  00E5  ;% 02   WB_R[TEMP2],WCTRL/PREV_CUR.ISCUR_WB       ;DCS ADD 02 SUBTEST #3
                                  00F0
                        00C00000  00F0  6$:      .LONG   ^X00C00000                        ;PSL DATA
                        07000000  00F4  7$:      .LONG   ^X07000000                        ;RESULTS SUBTEST #3
                                  00F8
                                  00F8  8$:
U 02, 7700,4800,5BE4,0300,7A70,1803  00F8  ;% 02   WB_R[TEMP0],WCTRL/IPL_WB                  ;DCS ADD 02 SUBTEST #1
                                  0103
                        CFDF00FF  0103  9$:      .LONG   ^XCFDF00FF                        ;MASK FOR CMPREGMSK PSEUDO-OP
                                  0107
                                  0107  BEGIN_CPU_DATA
                                  0002
                                  0002  DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 5700,4800,0364,0300,0070,1802  000C  ;% 01   WB_RDM,CCPSL/PSL_WB.CCBR_ALUS             ;DATA PATTERN TO PSL REG
U 02, 7700,4800,5BE4,0300,7A70,1803  0017  ;% 02   WB_R[TEMP0],WCTRL/IPL_WB                  ;PERORM TEST OF MICRO ORDER
U 03, 6700,C800,0364,0300,0870,1804  0022  ;% 03   RDM_WB,WB_PSL                             ;READ PSL TO RDM
U 04, 7300,4800,0364,0300,0470,1805  002D  ;% 04   NOP,STOP.CLOCK                            ;STOP CPU CLOCK
                                  0038
                                  0038  RTEMP_DATA
                                  0001
                        001F0000  0001          .LONG   ^X001F0000                        ;TEST PATTERN SUBTEST #1
                        00C00000  0005          .LONG   ^X00C00000                        ;TEST PATTERN SUBTEST #2
                        07C00000  0009          .LONG   ^X07C00000                        ;TEST PATTERN SUBTEST #3
                                  000D
                                  000D  END_CPU_DATA
                                  0107
                                  0107  END_TEST_DATA
                                  0107
                                  0107        ;-----------------------------------------------------------------------
                                  0107
                                  0107  ENDTEST
```

```
                        001B        .SBTTL  "TEST   02C     ALK Gate Array Test #1
                        001B  7743 ;*******************************************************************************
                        001B  7744 ;++
                        001B  7745 ;
                        001B  7746 ; FUNCTIONAL DESCRIPTION
                        001B  7747 ;
                        001B  7748 ;       This test consists of two subtests that check the circuitry associated
                        001B  7749 ;       with the ALUSO Flag in the ALK Gate Array.The first subtest executes
                        001B  7750 ;       seven ALPCTL Field patterns with the ALUSO Flag clear.The second sub-
                        001B  7751 ;       test executes nine ALPCTL Field patterns with the ALUSO Flag set.These
                        001B  7752 ;       patterns were derived from the XLOG Simulator and are reproduced after
                        001B  7753 ;       the test data.
                        001B  7754 ;
                        001B  7755 ; TEST ALGORITHM
                        001B  7756 ;
                        001B  7757 ;       SUBTEST 1
                        001B  7758 ;
                        001B  7759 ;       1. Load the micro words to perform the test.
                        001B  7760 ;       2. Execute the micro sequence.
                        001B  7761 ;               1. Clear the ALUSO Flag by shifting 0 right.
                        001B  7762 ;               2. Execute the test micro word.
                        001B  7763 ;               3. Read ALUSO Flag onto WBUS and into RDM.
                        001B  7764 ;       3. Check the results of the WBUS read.
                        001B  7765 ;
                        001B  7766 ;
                        001B  7767 ;       SUBTEST 2
                        001B  7768 ;
                        001B  7769 ;       1. Load the micro words to perform the test.
                        001B  7770 ;       2. Execute the micro sequence.
                        001B  7771 ;               1. Set the ALUSO Flag by shifting 1 right.
                        001B  7772 ;               2. Execute the test micro word.
                        001B  7773 ;               3. Read ALUSO Flag onto WBUS and into RDM.
                        001B  7774 ;       3. Check the results of the WBUS read.
                        001B  7775 ;
                        001B  7776 ; LOGIC DESCRIPTION
                        001B  7777 ;
                        001B  7778 ;       These tests,which use various data patterns, check the ALUSO Flag and
                        001B  7779 ;       associated circuitry within the ALK Gate Array. These data patterns are
                        001B  7780 ;       not shown as no repairs are possible to the gate array. It can only be
                        001B  7781 ;       replaced.
                        001B  7782 ;
                        001B  7783 ;       BOOLEAN MAP                 ALUSO_H
                        001B  7784 ;
                        001B  7785 ;       A 0000 0000 00 AA * A                                       A       A
                        001B  7786 ;       L PPPP PPPP PP LL * L                                       L       L
                        001B  7787 ;       U |||| |||| || UU * U                                       U       U
                        001B  7788 ;       S 9876 5432 10 03 * S                                       |       |
                        001B  7789 ;       O |||| |||| || 01 * O                                       M       L
                        001B  7790 ;       H HHHH HHHH HH || * |                                       S       S
                        001B  7791 ;                     HH * H  MUX     ALU     DQ                     B       B
                        001B  7792 ;       ***************************************************************************
                        001B  7793 ;     1 0 0000 1001 01 01 * 0  M.R     OR      Q_WX                   1       0
                        001B  7794 ;     2 0 0001 1110 01 10 * 0  M.R   ANDNOT    SQR                    0       1
                        001B  7795 ;     3 0 1000 1111 10 01 * 0  D.R   NOTAND    D_WX                   1       0
                        001B  7796 ;     4 0 1001 0000 10 10 * 0  D.R   A-B-CI    SQL.D_WX               0       1
```

```
001B  7797 ;     5 0 1011 1011 11 01 * 1  D.Q    AND.SL    SQR.D_WX          1       0
001B  7798 ;     6 0 1001 1011 11 11 * 0          DIVDS                     1       1
001B  7799 ;     7 0 1001 1111 10 11 * 1        DIVSLOW-                    1       1
001B  7800 ;
001B  7801 ;     1 1 0100 0110 10 10 * 1  M.S    A+B+CI.SR D_WX            0       1
001B  7802 ;     2 1 0110 0110 00 01 * 0 XM.Q    A+B+CI.SR NOP             1       0
001B  7803 ;     3 1 1001 0011 11 11 * 1  D.R    A-B-CI.SL SQR.WX_DSL.D_WX 1       1
001B  7804 ;     4 1 1001 1000 10 00 * 1  D.R      AND     SQL.D_WX        0       0
001B  7805 ;     5 1 1001 1110 10 01 * 1  D.R     ANDNOT   SQL.WX_D.D_WX   1       0
001B  7806 ;     6 1 1001 1110 11 01 * 0        MULSLOW+                    1       0
001B  7807 ;     7 1 1001 1111 01 10 * 1        MULFAST+                    0       1
001B  7808 ;     8 1 1001 1111 11 00 * 1         DIVDA                      0       0
001B  7809 ;
001B  7810 ; ASSUMPTIONS
001B  7811 ;
001B  7812 ;       This test requires no signals from any other CPU modules but assumes an
001B  7813 ;       operational RBUS, MBUS, WBUS and SBUS.
001B  7814 ;
001B  7815 ; ERROR DESCRIPTION
001B  7816 ;
001B  7817 ;   SUBTEST 1
001B  7818 ;
001B  7819 ;                                    Data expected ALUSO Flag<WB31>  Data received
001B  7820 ;       Iteration:      1                    0
001B  7821 ;                       2                    0
001B  7822 ;                       3                    0
001B  7823 ;                       4                    0
001B  7824 ;                       5                    1
001B  7825 ;                       6                    0
001B  7826 ;                       7                    1
001B  7827 ;
001B  7828 ;   SUBTEST 2
001B  7829 ;
001B  7830 ;                                    Data expected          Data received
001B  7831 ;       Iteration:      1                    1
001B  7832 ;                       2                    0
001B  7833 ;                       3                    1
001B  7834 ;                       4                    1
001B  7835 ;                       5                    1
001B  7836 ;                       6                    0
001B  7837 ;                       7                    1
001B  7838 ;                       8                    1
001B  7839 ;--
001B  7840 ;*******************************************************************************
001B  7841
001B  7842         SUBTEST 1
001E
001E       ;////////////////////////////////////////////////////////////////////////////
001E  7843
001E  7844         INITIALIZE                          ;DCLO low,Init Scratchpads
0020  7845         LOOP    I,1,7                       ;Set loop count
0027  7846         LDFIELD 8$,I,Byte,2$,64,68          ;Init Q Register
0038  7847         LDFIELD 9$,I,Byte,2$,34,39          ;Init D Register
0049  7848         LDFIELD 10$,I,Word,5$,48,57         ;Load ALPCTL Field of Test Micro Word
005A  7849         LDFIELD 11$,I,Byte,5$,64,68         ;Load MSRC Field of Test Micro Word
```

```
                        006B  7850          LDFIELD 12$,I,Byte,5$,34,39     ;Load RSRC Field of Test Micro Word
                        007C  7851          LOADDCS 1$,,0,3                 ;Load the test micro instrs.
                        0083  7852          LOADDCS 5$,,3,3                 ;Load the test micro instrs.
                        008A  7853          ERRLOOP                         ;Set the error loop
                        008C  7854          FETCH   0                       ;Get the first micro word
                        0091  7855          BRSTCLK 5                       ;Clock stop location 5
                        0095  7856          CMPREGMSK WHREG,7$,I,6$,,Long   ;Check the received data
                        00A1  7857          IFERROR ,<<ALK>>                ;Continue if no error else call out ALK
                        00A7  7858          ENDLOOP I                       ;Continue with loop
                        00AA  7859
                        00AA  7860          SUBTEST 2
                        00AD
                        00AD          ;////////////////////////////////////////////////////////////////////////////
                        00AD  7861
                        00AD  7862          LOOP    I,1,8                   ;Set loop count
                        00B4  7863          LDFIELD 14$,I,Byte,4$,64,68     ;Init Q Register
                        00C5  7864          LDFIELD 15$,I,Byte,4$,34,39     ;Init D Register
                        00D6  7865          LDFIELD 16$,I,Word,5$,48,57     ;Load ALPCTL Field of Test Micro Word
                        00E7  7866          LDFIELD 17$,I,Byte,5$,64,68     ;Load MSRC Field of Test Micro Word
                        00F8  7867          LDFIELD 18$,I,Byte,5$,34,39     ;Load RSRC Field of Test Micro Word
                        0109  7868          LOADDCS 3$,,0,3                 ;Load the test micro instrs.
                        0110  7869          LOADDCS 5$,,3,3                 ;Load the test micro instrs.
                        0117  7870          ERRLOOP                         ;Set the error loop
                        0119  7871          FETCH   0                       ;Get the first micro word
                        011E  7872          BRSTCLK 5                       ;Clock stop location 3
                        0122  7873          CMPREGMSK WHREG,13$,I,6$,,Long  ;Check the received data
                        012E  7874          IFERROR ,<<ALK>>                ;Continue if no error else call out ALK
                        0134  7875          ENDLOOP I                       ;Continue with loop
                        0137  7876          SKIP                            ;End of test
                        013E  7877
                        013E  7878 BEGIN_TEST_DATA
                        013E
                        013E          ;-----------------------------------------------------------------------------
                        013E  7879 ;+
                        013E  7880 ; The following micro words are used in these ALUSO tests of the ALK Gate Array
                        013E  7881 ;-
                        013E  7882
                        013E  7883 1$:
                        013E  7884 ;
                        013E  7885 ; Clear ALUSO flag-WBUS<31>
                        013E  7886 ;
U 00, 7700,C800,0364,0300,0470,1801     013E  7887 ;% 00    NOP                                 ;NOP Micro Instruction
U 01, 7700,8801,005C,0304,0470,1802     0149  7891 ;% 01    MUX/M.R2,ALU/A+B+CI.SL,RSRC/TEMP1,MSRC/TEMP1
                        0154  7895 ;
                        0154  7896 ; Load Q register and D register with conditioning data RSRC/$, MSRC/$
                        0154  7897 ;
                        0154  7898 2$:
U 02, 7700,4800,0057,0300,0470,1803     0154  7899 ;% 02    ALPCTL/WX_D_R.Q_M,RSRC/TEMP0,MSRC/TEMP0
                        015F  7903 ;
                        015F  7904 ; Set ALUSO flag - WBUS<31>
                        015F  7905 ;
                        015F  7906 3$:
U 00, 7700,C800,0364,0300,0470,1801     015F  7907 ;% 00    NOP
U 01, 7700,C801,005C,0318,0470,1802     016A  7911 ;% 01    MUX/M.R2,ALU/A+B+CI.SL,RSRC/TEMP6,MSRC/TEMP1
                        0175  7915 ;
```

```
                        0175  7916 ; Load Q register and D register with conditioning data RSRC/$, MSRC/$
                        0175  7917 ;
                        0175  7918 4$:
U 02, 7700,4800,0057,0300,0470,1803  0175  7919 ;% 02   ALPCTL/WX_D_R.Q_M,RSRC/TEMP0,MSRC/TEMP0
                        0180  7923 5$:
U 03, 7700,8800,5B64,0300,0470,1804  0180  7924 ;% 03   ALPCTL/NOP,ROT/ZERO              ;Test micro word
U 04, 6700,4800,037C,0300,0470,1805  018B  7928 ;% 04   ALPCTL/WB_ALUF,RDM_WB           ;Read flag into RDM
U 05, 7300,4800,0364,0300,0470,1806  0196  7932 ;% 05   NOP,STOP.CLOCK                  ;Stop clock and check results
                        01A1  7936
                80000000  01A1  7937 6$:     .Long   ^x80000000              ;Mask
                        01A5  7938
                00000000  01A5  7939 7$:     .Long      0                    ;Expected data SUBTEST 1
                00000000  01A9  7940         .Long      0
                00000000  01AD  7941         .Long      0
                00000000  01B1  7942         .Long      0
                80000000  01B5  7943         .Long   ^x80000000
                00000000  01B9  7944         .Long      0
                80000000  01BD  7945         .Long   ^x80000000
                        01C1  7946
                80000000  01C1  7947 13$:    .Long   ^x80000000              ;Expected data SUBTEST 2
                00000000  01C5  7948         .Long      0
                80000000  01C9  7949         .Long   ^x80000000
                80000000  01CD  7950         .Long   ^x80000000
                80000000  01D1  7951         .Long   ^x80000000
                00000000  01D5  7952         .Long      0
                80000000  01D9  7953         .Long   ^x80000000
                80000000  01DD  7954         .Long   ^x80000000
                        01E1  7955
                      01  01E1  7956 8$:     .Byte      1                    ;MSRC Field to load Q Register
                      01  01E2  7957         .Byte      1
                      01  01E3  7958         .Byte      1
                      01  01E4  7959         .Byte      1
                      06  01E5  7960         .Byte      6
                      01  01E6  7961         .Byte      1
                      01  01E7  7962         .Byte      1
                        01E8  7963
                      01  01E8  7964 14$:    .Byte      1                    ;MSRC Field to load Q Register
                      06  01E9  7965         .Byte      6
                      01  01EA  7966         .Byte      1
                      01  01EB  7967         .Byte      1
                      01  01EC  7968         .Byte      1
                      01  01ED  7969         .Byte      1
                      01  01EE  7970         .Byte      1
                      01  01EF  7971         .Byte      1
                        01F0  7972
                      01  01F0  7973 9$:     .Byte      1                    ;RSRC Field to load D Register
                      01  01F1  7974         .Byte      1
                      05  01F2  7975         .Byte      5
                      02  01F3  7976         .Byte      2
                      06  01F4  7977         .Byte      6
                      07  01F5  7978         .Byte      7
                      06  01F6  7979         .Byte      6
                        01F7  7980
                      01  01F7  7981 15$:    .Byte      1                    ;RSRC Field to load D Register
                      01  01F8  7982         .Byte      1
```

```
                07  01F9  7983          .Byte     7
                01  01FA  7984          .Byte     1
                06  01FB  7985          .Byte     6
                06  01FC  7986          .Byte     6
                04  01FD  7987          .Byte     4
                07  01FE  7988          .Byte     7
                    01FF  7989
              0025  01FF  7990 10$:      .Word     ^x025                ;ALPCTL field
              0079  0201  7991          .Word     ^x079
              023E  0203  7992          .Word     ^x23E
              0242  0205  7993          .Word     ^x242
              02EF  0207  7994          .Word     ^x2EF
              026F  0209  7995          .Word     ^x26F
              027E  020B  7996          .Word     ^x27E
                    020D  7997
              011A  020D  7998 16$:      .Word     ^x11A                ;ALPCTL field
              0198  020F  7999          .Word     ^x198
              024F  0211  8000          .Word     ^x24F
              0262  0213  8001          .Word     ^x262
              027A  0215  8002          .Word     ^x27A
              027B  0217  8003          .Word     ^x27B
              027D  0219  8004          .Word     ^x27D
              027F  021B  8005          .Word     ^x27F
                    021D  8006
                01  021D  8007 11$:      .Byte     1                    ;MSRC Field for test micro word
                02  021E  8008          .Byte     2
                01  021F  8009          .Byte     1
                01  0220  8010          .Byte     1
                01  0221  8011          .Byte     1
                01  0222  8012          .Byte     1
                01  0223  8013          .Byte     1
                    0224  8014
                02  0224  8015 17$:      .Byte     2                    ;MSRC Field for test micro word
                02  0225  8016          .Byte     2
                01  0226  8017          .Byte     1
                01  0227  8018          .Byte     1
                01  0228  8019          .Byte     1
                01  0229  8020          .Byte     1
                01  022A  8021          .Byte     1
                01  022B  8022          .Byte     1
                    022C  8023
                06  022C  8024 12$:      .Byte     6                    ;RSRC Field for test micro word
                04  022D  8025          .Byte     4
                06  022E  8026          .Byte     6
                01  022F  8027          .Byte     1
                01  0230  8028          .Byte     1
                07  0231  8029          .Byte     7
                07  0232  8030          .Byte     7
                    0233  8031
                01  0233  8032 18$:      .Byte     1                    ;RSRC Field for test micro word
                01  0234  8033          .Byte     1
                02  0235  8034          .Byte     2
                01  0236  8035          .Byte     1
                05  0237  8036          .Byte     5
                05  0238  8037          .Byte     5
```

```
                    02  0239  8038          .Byte      2
                    01  023A  8039          .Byte      1
                        023B  8040
                        023B  8041 BEGIN_CPU_DATA
                        0002  8042
                        0002  8043 RTEMP_DATA
                        0001  8044
          00000000      0001  8045          .Long      0               ;TEMP0 - IGNORE
          00000000      0005  8046          .Long      0               ;TEMP1
          00000001      0009  8047          .Long      1               ;TEMP2
          C0000000      000D  8048          .Long      ^xC0000000      ;TEMP3
          FFFFFFFE      0011  8049          .Long      ^xFFFFFFFE      ;TEMP4
          7FFFFFFF      0015  8050          .Long      ^x7FFFFFFF      ;TEMP5
          80000000      0019  8051          .Long      ^x80000000      ;TEMP6
          FFFFFFFF      001D  8052          .Long      ^xFFFFFFFF      ;TEMP7
                        0021  8053
                        0021  8054 END_CPU_DATA
                        023B  8055
                        023B  8056 END_TEST_DATA
                        023B
                        023B          ;-----------------------------------------------------------------------------
                        023B  8057
                        023B  8058 ENDTEST
```

```
001B        .SBTTL  "TEST   02D     ALK Gate Array Test #2
001B  8060 ;*******************************************************************************
001B  8061 ;++
001B  8062 ;
001B  8063 ; FUNCTIONAL DESCRIPTION
001B  8064 ;
001B  8065 ;       This test consists of two subtests that check the circuitry associated
001B  8066 ;       with the ALKC Flag in the ALK Gate Array.The first subtest executes
001B  8067 ;       eight ALPCTL Field patterns with the ALKC Flag clear.The second subtest
001B  8068 ;       executes ten ALPCTL Field patterns with the ALKC Flag set.These
001B  8069 ;       patterns were derived from the XLOG Simulator and are reproduced after
001B  8070 ;       the test data.
001B  8071 ;
001B  8072 ; TEST ALGORITHM
001B  8073 ;
001B  8074 ;       SUBTEST 1
001B  8075 ;
001B  8076 ;       1. Load the micro words to perform the test.
001B  8077 ;       2. Execute the micro sequence.
001B  8078 ;               1. Clear the ALKC Flag by adding zero to zero.
001B  8079 ;               2. Execute the test micro word.
001B  8080 ;               3. Read ALKC Flag onto WBUS and into RDM.
001B  8081 ;       3. Check the results of the WBUS read.
001B  8082 ;
001B  8083 ;
001B  8084 ;       SUBTEST 2
001B  8085 ;
001B  8086 ;       1. Load the micro words to perform the test.
001B  8087 ;       2. Execute the micro sequence.
001B  8088 ;               1. Set the ALKC Flag by adding C0000000 to C0000000.
001B  8089 ;               2. Execute the test micro word.
001B  8090 ;               3. Read ALKC Flag onto WBUS and into RDM.
001B  8091 ;       3. Check the results of the WBUS read.
001B  8092 ;
001B  8093 ;  DATA PATTERNS
001B  8094 ;
001B  8095 ;       SUBTEST 1
001B  8096 ;
001B  8097 ;    MUX    ALU       D Register     Q Register    RTEMP    MTEMP     ROTATE
001B  8098 ;
001B  8099 ;1  XM.R   B-A-CI          1              0           4        1        ZERO
001B  8100 ;2   D.R   ANDNOT         1              0           4        1        ZERO
001B  8101 ;3   D.S   B-A-CI         0              0           1        1        ZERO
001B  8102 ;4   0.S   A-B-CI         0              0           2        1        RL.RR.P
001B  8103 ;5   0.S   B-A-CI         0              0           2        1        RL.RR.P
001B  8104 ;6   M.R   AND.SR         FFFFFFFE       0           1        1        ZERO
001B  8105 ;7   D.R   A+B+CI.SL      1              0           1        1        ZERO
001B  8106 ;
001B  8107 ;       SUBTEST 2
001B  8108 ;
001B  8109 ;    MUX     ALU       D Register     Q Register    RTEMP     MTEMP    ROTATE
001B  8110 ;
001B  8111 ;1  M.S   A+B+CI          0              0           1         1       ZERO
001B  8112 ;2  M.S    XOR            0              0           1         1       ZERO
001B  8113 ;3  D.R    AND            1              0           2         1       ZERO
```

```
001B  8114 ;4  D.R  AND.SR        1            0          2          1     ZERO
001B  8115 ;5  D.R  B-A-CI        0            0          1          1     ZERO
001B  8116 ;6  D.R  ANDNOT        0            0          7          1     ZERO
001B  8117 ;7  D.R  NOTAND        FFFFFFFF     0          1          1     ZERO
001B  8118 ;8  R.S  B-A-CI        0            0          1          1     ZERO
001B  8119 ;9   WX<-SBUS          0            0          7          1     ZERO
001B  8120 ;
001B  8121 ; LOGIC DESCRIPTION
001B  8122 ;
001B  8123 ;       These tests check the ALKC Flag and associated circuitry within the
001B  8124 ;       ALK Gate Array.
001B  8125 ;
001B  8126 ;          BOOLEAN MAP          ALKC_H
001B  8127 ;
001B  8128 ;       AA 0000 0000 0A * A
001B  8129 ;       LL PPPP PPPP PL * L
001B  8130 ;       KU |||| |||| |U * K
001B  8131 ;       CC 9876 5432 00 * C
001B  8132 ;       H| |||| |||| |0 * |
001B  8133 ;        H HHHH HHHH H| * H
001B  8134 ;                     H *   MUX    ALU        ALU<00>
001B  8135 ;       ******************************************
001B  8136 ;     1 00 0101 1100 11 * 0 XM.R  B-A-CI        1
001B  8137 ;     2 00 1000 1110 11 * 0  D.R  ANDNOT        1
001B  8138 ;     3 00 1100 1100 00 * 0  D.S  B-A-CI        0
001B  8139 ;     4 00 1101 0000 01 * 1  0.S  A-B-CI        1
001B  8140 ;     5 00 1101 1100 11 * 0     WX<-S           1
001B  8141 ;     6 01 1001 0010 11 * 0  D.R A-B-CI.SR      1
001B  8142 ;     7 01 1010 0111 00 * 0  D.0 A+B+CI.SL      0
001B  8143 ;
001B  8144 ;     1 10 0100 0100 10 * 0  M.S  A+B+CI        0
001B  8145 ;     2 10 0100 1101 10 * 1  M.S   XOR          0
001B  8146 ;     3 10 1001 1000 11 * 1  D.R   AND          1
001B  8147 ;     4 10 1001 1010 11 * 0  D.R  AND.SR        1
001B  8148 ;     5 10 1001 1100 00 * 1  D.R  B-A-CI        0
001B  8149 ;     6 10 1001 1110 00 * 1  D.R  ANDNOT        0
001B  8150 ;     7 10 1001 1111 10 * 1  D.R  NOTAND        0
001B  8151 ;     8 11 1111 1100 11 * 0  R.S  B-A-CI        1
001B  8152 ;     9 11 1101 1100 11 * 1    WX<-S            1
001B  8153 ;
001B  8154 ; ASSUMPTIONS
001B  8155 ;
001B  8156 ;       This test requires no signals from any other CPU modules but assumes an
001B  8157 ;       operational RBUS, MBUS, WBUS and SBUS.
001B  8158 ;
001B  8159 ; ERROR DESCRIPTION
001B  8160 ;
001B  8161 ;   SUBTEST 1
001B  8162 ;
001B  8163 ;                                Data expected          Data received
001B  8164 ;       Iteration:      1              0
001B  8165 ;                       2              0
001B  8166 ;                       3              0
001B  8167 ;                       4              40000000
001B  8168 ;                       5              0
```

```
001B  ;                          6                0
001B  ;                          7                0
001B  ;
001B  ;   SUBTEST 2
001B  ;
001B  ;                                Data expected          Data received
001B  ;       Iteration:         1                0
001B  ;                          2                40000000
001B  ;                          3                40000000
001B  ;                          4                0
001B  ;                          5                40000000
001B  ;                          6                40000000
001B  ;                          7                40000000
001B  ;                          8                0
001B  ;                          9                40000000
001B  ;--
001B  ;*****************************************************************************
001B
001B          SUBTEST 1
001E
001E      ;////////////////////////////////////////////////////////////////////////
001E
001E          INITIALIZE                          ;DCLO low,Init Scratchpads
0020          LOOP    I,1,7                       ;Set loop count
0027          LDFIELD 8$,,Byte,2$,64,68           ;Init Q Register
0038          LDFIELD 9$,I,Byte,2$,34,39          ;Init D Register
0049          LDFIELD 10$,I,Word,5$,48,57         ;Load ALPCTL Field of Test Micro Word
005A          LDFIELD 11$,I,Byte,5$,58,63         ;Load ROTATE Field of Test Micro Word
006B          LDFIELD 12$,I,Byte,5$,64,68         ;Load MSRC Field of Test Micro Word
007C          LDFIELD 13$,I,Byte,5$,34,39         ;Load RSRC Field of Test Micro Word
008D          LOADDCS 1$,,0,4                     ;Load the test micro instrs.
0094          LOADDCS 5$,,4,3                     ;Load the test micro instrs.
009B          ERRLOOP                             ;Set the error loop
009D          FETCH   0                           ;Get the first micro word
00A2          BRSTCLK 6                           ;Clock stop location 6
00A6          CMPREGMSK WHREG,7$,I,6$,,Long       ;Check the received data
00B2          IFERROR ,<<ALK>>                    ;Continue if no error else call out ALK
00B8          ENDLOOP I                           ;Continue with loop
00BB
00BB          SUBTEST 2
00BE
00BE      ;////////////////////////////////////////////////////////////////////////
00BE
00BE          LOOP    I,1,9                       ;Set loop count
00C5          LDFIELD 8$,,Byte,4$,64,68           ;Init Q Register
00D6          LDFIELD 16$,I,Byte,4$,34,39         ;Init D Register
00E7          LDFIELD 17$,I,Word,5$,48,57         ;Load ALPCTL Field of Test Micro Word
00F8          LDFIELD 18$,I,Byte,5$,58,63         ;Load ROTATE Field of Test Micro Word
0109          LDFIELD 12$,,Byte,5$,64,68          ;Load MSRC Field of Test Micro Word
011A          LDFIELD 20$,I,Byte,5$,34,39         ;Load RSRC Field of Test Micro Word
012B          LOADDCS 3$,,0,7                     ;Load the test micro instrs.
0132          ERRLOOP                             ;Set the error loop
0134          FETCH   0                           ;Get the first micro word
0139          BRSTCLK 6                           ;Clock stop location 6
013D          CMPREGMSK WHREG,14$,I,6$,,Long      ;Check the received data
```

```
                                0149  8220          IFERROR ,<<ALK>>                ;Continue if no error else call out ALK
                                014F  8221          ENDLOOP I                       ;Continue with loop
                                0152  8222          SKIP                            ;End of test
                                0159  8223
                                0159  8224 BEGIN_TEST_DATA
                                0159
                                0159          ;-----------------------------------------------------------------------
                                0159  8225 ;+
                                0159  8226 ; The following micro words are used in these ALKC tests of the ALK Gate Array
                                0159  8227 ;-
                                0159  8228 ;
                                0159  8229 ; Clear position latch then clear ALKC flag - WBUS<31>
                                0159  8230 ;
                                0159  8231 1$:
U 00, 7700,C800,0364,0300,0470,1801   0159  8232 ;% 00    NOP
U 01, 7700,8800,BC47,0304,0470,1802   0164  8236 ;% 01    ALPCTL/WX_R.Q_M,ROT/SL.PL_WB,RSRC/TEMP1
U 02, 7700,0801,0050,0304,0470,1803   016F  8240 ;% 02    MUX/M.R2,ALU/A+B+CI,RSRC/TEMP1,MSRC/TEMP1
                                017A  8244 ;
                                017A  8245 ; Load Q register and D register with conditioning data RSRC/8$, MSRC/9$
                                017A  8246 ;
                                017A  8247 2$:
U 03, 7700,C800,0057,0300,0470,1804   017A  8248 ;% 03    ALPCTL/WX_D_R.Q_M,RSRC/TEMP0,MSRC/TEMP0
                                0185  8252 ;
                                0185  8253 ; Clear position latch
                                0185  8254 ;
                                0185  8255 3$:
U 00, 7700,C800,0364,0300,0470,1801   0185  8256 ;% 00    NOP
U 01, 7700,8800,BC47,0304,0470,1802   0190  8260 ;% 01    ALPCTL/WX_R.Q_M,ROT/SL.PL_WB,RSRC/TEMP1
                                019B  8264 ;
                                019B  8265 ; Load Q register and D register with conditioning data RSRC/8$, MSRC/16$
                                019B  8266 ;
                                019B  8267 4$:
U 02, 7700,4800,0057,0300,0470,1803   019B  8268 ;% 02    ALPCTL/WX_D_R.Q_M,RSRC/TEMP0,MSRC/TEMP0
                                01A6  8272 ;
                                01A6  8273 ; Set ALKC flag - WBUS<30>
                                01A6  8274 ;
U 03, 7700,C803,0010,030C,0470,1804   01A6  8275 ;% 03    MUX/M.R1,ALU/A+B+CI,RSRC/TEMP3,MSRC/TEMP3,DQ1/NOP
                                01B1  8279 5$:
U 04, 7700,0800,5B64,0300,0470,1805   01B1  8280 ;% 04    ALPCTL/NOP,ROT/ZERO     ;Test micro word
U 05, 6700,4800,037C,0300,0470,1806   01BC  8284 ;% 05    ALPCTL/WB_ALUF,RDM_WB   ;Read flag into RDM
U 06, 7300,C800,0364,0300,0470,1807   01C7  8288 ;% 06    NOP,STOP.CLOCK          ;Stop clock and check results
                                01D2  8292
                   40000000     01D2  8293 6$:      .Long   ^x40000000      ;Mask
                                01D6  8294
                   00000000     01D6  8295 7$:      .Long      0            ;Expected data - SUBTEST 1
                   00000000     01DA  8296          .Long      0
                   00000000     01DE  8297          .Long      0
                   40000000     01E2  8298          .Long   ^x40000000
                   00000000     01E6  8299          .Long      0
                   00000000     01EA  8300          .Long      0
                   00000000     01EE  8301          .Long      0
                                01F2  8302
                   00000000     01F2  8303 14$:     .Long      0            ;Expected data - SUBTEST 2
                   40000000     01F6  8304          .Long   ^x40000000
                   40000000     01FA  8305          .Long   ^x40000000
```

```
00000000  01FE  8306          .Long     0
40000000  0202  8307          .Long   ^x40000000
40000000  0206  8308          .Long   ^x40000000
40000000  020A  8309          .Long   ^x40000000
00000000  020E  8310          .Long     0
40000000  0212  8311          .Long   ^x40000000
          0216  8312
      01  0216  8313 8$:      .Byte     1           ;MSRC Field to load Q Register - SUBTEST 1,2
          0217  8314
      02  0217  8315 9$:      .Byte     2           ;RSRC Field to load D Register - SUBTEST 1
      02  0218  8316          .Byte     2
      01  0219  8317          .Byte     1
      01  021A  8318          .Byte     1
      01  021B  8319          .Byte     1
      04  021C  8320          .Byte     4
      02  021D  8321          .Byte     2
          021E  8322
      01  021E  8323 16$:     .Byte     1           ;RSRC Field to load D Register - SUBTEST 2
      01  021F  8324          .Byte     1
      02  0220  8325          .Byte     2
      02  0221  8326          .Byte     2
      01  0222  8327          .Byte     1
      01  0223  8328          .Byte     1
      07  0224  8329          .Byte     7
      01  0225  8330          .Byte     1
      01  0226  8331          .Byte     1
          0227  8332
0171  0227  8333 10$:     .Word   ^x171           ;ALPCTL field - SUBTEST 1
0239  0229  8334          .Word   ^x239
0330  022B  8335          .Word   ^x330
0340  022D  8336          .Word   ^x340
0371  022F  8337          .Word   ^x371
0249  0231  8338          .Word   ^x249
029C  0233  8339          .Word   ^x29C
          0235  8340
0111  0235  8341 17$:     .Word   ^x111           ;ALPCTL field - SUBTEST 2
0135  0237  8342          .Word   ^x135
0261  0239  8343          .Word   ^x261
0269  023B  8344          .Word   ^x269
0270  023D  8345          .Word   ^x270
0278  023F  8346          .Word   ^x278
027D  0241  8347          .Word   ^x27D
03F1  0243  8348          .Word   ^x3F1
0371  0245  8349          .Word   ^x371
          0247  8350
  16  0247  8351 11$:     .Byte   ^x16            ;ROTATE Field for test micro word - SUBTEST 1
  16  0248  8352          .Byte   ^x16
  16  0249  8353          .Byte   ^x16
  22  024A  8354          .Byte   ^x22
  22  024B  8355          .Byte   ^x22
  16  024C  8356          .Byte   ^x16
  16  024D  8357          .Byte   ^x16
  16  024E  8358          .Byte   ^x16
          024F  8359
  16  024F  8360 18$:     .Byte   ^x16            ;ROTATE Field for test micro word - SUBTEST 2
```

```
                     16  0250  8361          .Byte   ^x16
                     16  0251  8362          .Byte   ^x16
                     16  0252  8363          .Byte   ^x16
                     16  0253  8364          .Byte   ^x16
                     16  0254  8365          .Byte   ^x16
                     16  0255  8366          .Byte   ^x16
                     16  0256  8367          .Byte   ^x16
                     22  0257  8368          .Byte   ^x22
                         0258  8369
                     01  0258  8370 12$:     .Byte     1           ;MSRC Field for test micro word - SUBTEST 1,2
                         0259  8371
                     04  0259  8372 13$:     .Byte     4           ;RSRC Field for test micro word - SUBTEST 1
                     04  025A  8373          .Byte     4
                     01  025B  8374          .Byte     1
                     02  025C  8375          .Byte     2
                     02  025D  8376          .Byte     2
                     01  025E  8377          .Byte     1
                     01  025F  8378          .Byte     1
                         0260  8379
                     01  0260  8380 20$:     .Byte     1           ;RSRC Field for test micro word - SUBTEST 2
                     01  0261  8381          .Byte     1
                     02  0262  8382          .Byte     2
                     02  0263  8383          .Byte     2
                     01  0264  8384          .Byte     1
                     07  0265  8385          .Byte     7
                     01  0266  8386          .Byte     1
                     01  0267  8387          .Byte     1
                     07  0268  8388          .Byte     7
                         0269  8389
                         0269  8390 BEGIN_CPU_DATA
                         0002  8391
                         0002  8392 RTEMP_DATA
                         0001  8393
00000000                 0001  8394          .Long     0           ;TEMP0 - IGNORE
00000000                 0005  8395          .Long     0           ;TEMP1
00000001                 0009  8396          .Long     1           ;TEMP2
C0000000                 000D  8397          .Long   ^xC0000000    ;TEMP3
FFFFFFFE                 0011  8398          .Long   ^xFFFFFFFE    ;TEMP4
7FFFFFFF                 0015  8399          .Long   ^x7FFFFFFF    ;TEMP5
80000000                 0019  8400          .Long   ^x80000000    ;TEMP6
FFFFFFFF                 001D  8401          .Long   ^xFFFFFFFF    ;TEMP7
00000002                 0021  8402          .Long     2           ;TEMP8
                         0025  8403
                         0025  8404 END_CPU_DATA
                         0269  8405
                         0269  8406 END_TEST_DATA
                         0269
                         0269        ;-----------------------------------------------------------------------------
                         0269  8407
                         0269  8408 ENDTEST
```

```
001B        .SBTTL  "TEST   02E     ALK Gate Array Test #3
001B  8410 ;*******************************************************************************
001B  8411 ;++
001B  8412 ;
001B  8413 ; FUNCTIONAL DESCRIPTION
001B  8414 ;
001B  8415 ;       This test, which consists of three subtests, checks the ALU shift and
001B  8416 ;       Q Register shift functions of the ALK Gate Array Chip. This test has no
001B  8417 ;       ALU Flags set.
001B  8418 ;
001B  8419 ; TEST ALGORITHM
001B  8420 ;
001B  8421 ;       SUBTEST 1
001B  8422 ;
001B  8423 ;       1. Load the micro words to perform the test.
001B  8424 ;       2. Setup a loop of eighteen iterations.
001B  8425 ;       3. Load appropriate micro fields.
001B  8426 ;       4. Execute the micro sequence.
001B  8427 ;               1. Condition the DPM with data from Tables 1 thru .
001B  8428 ;               2. Test the DPM with data from Tables 1 thru .
001B  8429 ;               3. Read WBUS into RDM.
001B  8430 ;       5. Check the results of the WBUS read.
001B  8431 ;       6. Looped eighteen times? If not go to step three.
001B  8432 ;
001B  8433 ;       SUBTEST 2
001B  8434 ;
001B  8435 ;       1. Load the micro words to perform the test.
001B  8436 ;       2. Setup a loop of eighteen iterations.
001B  8437 ;       3. Load appropriate micro fields.
001B  8438 ;       4. Execute the micro sequence.
001B  8439 ;               1. Condition the DPM with data from Tables 1 thru .
001B  8440 ;               2. Test the DPM with data from Tables 1 thru .
001B  8441 ;               3. Read WBUS into RDM.
001B  8442 ;       5. Check the results of the WBUS read (Q Register).
001B  8443 ;       6. Looped eighteen times? If not go to step three.
001B  8444 ;
001B  8445 ;       SUBTEST 3
001B  8446 ;
001B  8447 ;       1. Load the micro words to perform the test.
001B  8448 ;       2. Setup a loop of eighteen iterations.
001B  8449 ;       3. Load appropriate micro fields.
001B  8450 ;       4. Execute the micro sequence.
001B  8451 ;               1. Condition the DPM with data from Tables 1 thru .
001B  8452 ;               2. Test the DPM with data from Tables 1 thru .
001B  8453 ;               3. Read WBUS into RDM.
001B  8454 ;       5. Check the results of the WBUS read (D Register).
001B  8455 ;       6. Looped eighteen times? If not go to step three.
001B  8456 ;
001B  8457 ; DATA PATTERNS
001B  8458 ;
001B  8459 ; Including the initial load the D and Q Registers
001B  8460 ;
001B  8461 ;       AMUX            BMUX            QREG            DREG
001B  8462 ;
001B  8463 ;1      80000001        0               0               0
```

```
001B  8464 ;2      0                 80000001          0                 0
001B  8465 ;3      80000001          0                 0                 0
001B  8466 ;4      80000001          80000001          1                 0
001B  8467 ;5      1                 80000000          80000000          0
001B  8468 ;6      80000001          1                 80000001          80000001
001B  8469 ;7      80000001          0                 80000001          0
001B  8470 ;8      80000000          0                 1                 80000000
001B  8471 ;9      80000000          80000001          80000001          80000000
001B  8472 ;10     80000000          80000001          80000001          80000000
001B  8473 ;11     0                 80000001          80000001          0
001B  8474 ;12     80000000          80000000          0                 0
001B  8475 ;13     80000001          80000001          80000001          0
001B  8476 ;14     0                 0                 1                 0
001B  8477 ;15     80000000          7FFFFFFF          80000001          80000000
001B  8478 ;16     0                 0                 0                 0
001B  8479 ;17     80000001          0                 0                 80000001
001B  8480 ;18     80000000          80000000          1                 80000000
001B  8481 ;
001B  8482 ; LOGIC DESCRIPTION
001B  8483 ;
001B  8484 ;This test checks the ALK Gate Array's gates for proper operation. These
001B  8485 ;patterns were developed using the XLOG Simulator.
001B  8486 ;
001B  8487 ;The following display is the simulator output,with additions to make it more
001B  8488 ;legible. The AMUX and BMUX represent the two inputs to the ALU. The PSLC_H
001B  8489 ;input represents the PSL 'C' bit, QMSB_H represents the Most Significant Bit
001B  8490 ;of the Q Register while QR00_H represents the LSB of the Q Register. ALU31_H
001B  8491 ;represents the MSB of the ALU and ALU00_H equals the LSB of the ALU. ROT_5_H
001B  8492 ;thru ROT_0_H are the ROTATE Field of the Micro Word. OP_9_H thru OP_0_H are
001B  8493 ;the ALPCTL Field of the Micro Word. WMUX_CONT indicates whether the WMUX
001B  8494 ;Tri-State Drivers are active for any given test. MUX, ALU and DQ CONT is the
001B  8495 ;decode of the simulator output recognizable by the Assembler. The two columns
001B  8496 ;under the ROTATE heading decode the ROTATE Field output according to the DPM
001B  8497 ;Functional Specification. The line showing ALU and some data and QREG and some
001B  8498 ;data represent the necessary data in both entities to promote a successful and
001B  8499 ;meaningful test for that particular OPCODE and ROTATE Field.
001B  8500 ;
001B  8501 ; NO FLAGS ARE SET AND OUTPUT IS ENABLED
001B  8502 ;
001B  8503 ;                 BOOLEAN MAP
001B  8504 ;
001B  8505 ;DD ALP QQR RRR RR 0000 0000 00 AA * AAQQQQ W
001B  8506 ;SS LOS MRO 000 00 PPPP PPPP PP LL * SSSSSS M
001B  8507 ;II UOL S0T TTT TT |||| |||| || UU * IIIIII U
001B  8508 ;ZZ SPC B0| ||| || 9876 5432 10 30 * 000000 X
001B  8509 ;EE 0|| ||5 432 10 |||| |||| || 10 * |||||| 
001B  8510 ;|| |HH HH| ||| || HHHH HHHH HH || * 303100 C
001B  8511 ;10 H     H HHH HH                HH * 101570 O
001B  8512 ;||                                 * |||||| N                          ROTATE
001B  8513 ;HH                                 * LLLLLL T MUX   ALU     DQCON      ASI/QSI
001B  8514 ;*************************************************************************
001B  8515 ;
001B  8516 ;1
001B  8517 ;DSIZE_1_H 0    DSIZE_0_H 0     ALUSO_H 0
001B  8518 ;AMUX(M5)_80000001      BMUX(R1)_0      CI=1    LOOP_H 0
```

```
001B  8519 ;00 000 000 010 10 0001 0011 11 10 * 111111 E M.R2 A-B-CI.SL SQR.D_WX      SHF
001B  8520 ;ALU 80000000   QREG 0  PSL<C>=0        RESULTS: WB=0, QREG=0, DREG=0
001B  8521 ;
001B  8522 ;2
001B  8523 ;DSIZE_1_H 0    DSIZE_0_H 0     ALUSO_H 0
001B  8524 ;AMUX(M1)_0     BMUX(R5)_80000001       CI=PSL<C>=0     LOOP_H  0
001B  8525 ;00 000 000 011 11 0101 0111 11 11 * 101111 E XM.R A+B+CI.SL Q_WX.D_WX     ROT
001B  8526 ;ALU 80000001   QREG 0  PSL<C> 0        RESULTS: WB=3, QREG=3, DREG=3
001B  8527 ;
001B  8528 ;3
001B  8529 ;DSIZE_1_H 0            DSIZE_0_H 0     ALUSO_H 0
001B  8530 ;AMUX(M5)_80000001      BMUX(S)_0       CI=PSL<C>=0     LOOP_H  0
001B  8531 ;00 000 000 011 11 0100 0010 00 11 * 111111 E M.S A-B-CI.SR     NOP     ROT
001B  8532 ;ALU 80000001   QREG 0  PSL<C> 0        WB=40000000     QREG 0  DREG 0
001B  8533 ;
001B  8534 ;4
001B  8535 ;DSIZE_1_H 0    DSIZE_0_H 0     ALUSO_H 0
001B  8536 ;AMUX(M5)_80000001      BMUX(R5)_80000001       CI=PSL<C>=0     LOOP_H  0
001B  8537 ;00 000 011 010 11 0001 1011 11 11 * 101111 E M.R2 (A.AND.B).SL SQR.D_WX   SHF
001B  8538 ;ALU 80000001   QREG 1  PSL<C> 0        RESULTS: WB=3, QREG=0, DREG=3
001B  8539 ;
001B  8540 ;5
001B  8541 ;DSIZE_1_H 0    DSIZE_0_H 0     ALUSO_H 0
001B  8542 ;AMUX(M2)_1     BMUX(Q)_80000000        CI=0    LOOP_H  0
001B  8543 ;00 000 100 011 00 0011 0101 01 01 * 111111 E M.Q2 A+B+CI.BCD   SQR         ROT
001B  8544 ;ALU 1  QREG 80000000   PSL<C> 0 RESULTS: WB=80000001, QREG=40000000, DREG=0
001B  8545 ;
001B  8546 ;6
001B  8547 ;DSIZE_1_H 0    DSIZE_0_H 0     ALUSO_H 0
001B  8548 ;AMUX(D)_80000001       BMUX(R2)_1      CI=0    LOOP_H  0
001B  8549 ;00 001 110 100 00 1001 0011 11 10 * 110111 E D.R2 A-B-CI.SL SQR.WX_DSL.D_WX
001B  8550 ;ALU 80000000   QREG 80000001   PSL<C> 1 RESULTS: WB=2, QREG=C0000000, DREG=2
001B  8551 ;
001B  8552 ;7                                                                       0   1
001B  8553 ;DSIZE_1_H 0    DSIZE_0_H 0     ALUSO_H 0
001B  8554 ;AMUX(M5)_80000001      BMUX(S)_0       CI=1    LOOP_H  0
001B  8555 ;00 001 110 101 10 0111 0010 01 10 * 111111 E XM.S A-B-CI.SR Q_WX       1   0
001B  8556 ;ALU 80000000   QREG 80000001   PSL<C> 1  RESULTS: WB=0 QREG 0  DREG 0
001B  8557 ;
001B  8558 ;8
001B  8559 ;DSIZE_1_H 0    DSIZE_0_H 0     ALUSO_H 1
001B  8560 ;AMUX(D)_80000000       BMUX(R1)_0      CI=PSL<C>=1     LOOP_H  0
001B  8561 ;00 101 010 011 11 1000 0110 10 11 * 011111 E D.R1 A+B+CI.SR    D_WX        ROT
001B  8562 ;ALU 80000001   QREG 1  PSL<C> 1    RESULTS: WB=C0000000, QREG=1, DREG=C0000000
001B  8563 ;
001B  8564 ;9
001B  8565 ;DSIZE_1_H 0    DSIZE_0_H 1     ALUSO_H 0
001B  8566 ;AMUX(D)_80000000       BMUX(Q)_80000001        CI=ALKC=0       LOOP_H  0
001B  8567 ;01 000 110 010 01 1011 1010 11 10 * 111111 E D.Q2 (A.AND.B).SR SQR.D_WX   SHF
001B  8568 ;ALU 80000000   QREG 80000001   PSL<C> 0        RESULTS: WB=40000000,
001B  8569 ;                                            RESULTS: QREG=40000000, DREG=40000000
001B  8570 ;10
001B  8571 ;DSIZE_1_H 0    DSIZE_0_H 1     ALUSO_H 0
001B  8572 ;AMUX(D)_80000000       BMUX(Q)_80000001        CI=ALKC=1       LOOP_H  0
001B  8573 ;01 000 110 010 01 1011 1010 11 10 * 111111 E D.Q2 (A.AND.B).SR SQR.D_WX   SHF
```

```
001B  8574 ;ALU 80000000   QREG 80000001   PSL<C> 0        RESULTS: WB=40000000,
001B  8575 ;                                          RESULTS: QREG=40000000, DREG=40000000
001B  8576 ;11
001B  8577 ;DSIZE_1_H 1    DSIZE_0_H 0 ALUSO_H 0
001B  8578 ;AMUX(M1)_0     BMUX(Q)_80000001        CI=0    LOOP_H  0
001B  8579 ;10 001 111 011 00 0011 1011 10 00 * 101111 E M.Q2 (A.AND.B).SL SQL.D_WX    ROT
001B  8580 ;ALU 0  QREG 80000001   PSL<C> 1        RESULTS: WB=1, QREG=2, DREG=1
001B  8581 ;
001B  8582 ;12
001B  8583 ;DSIZE_1_H 1    DSIZE_0_H 1     ALUSO_H 0
001B  8584 ;AMUX(M4)_80000000      BMUX(R4)_80000000       CI=0    LOOP_H  0
001B  8585 ;11 000 000 011 00 0001 1011 01 10 * 110111 E M.R2 (A.AND.B).SL SQR         ROT
001B  8586 ;ALU 80000000   QREG 0  PSL<C> 0        RESULTS: WB=0, QREG=80000000, DREG=0
001B  8587 ;
001B  8588 ;13
001B  8589 ;DSIZE_1_H 1    DSIZE_0_H 1     ALUSO_H 0
001B  8590 ;AMUX(R5)_80000001      BMUX(Q)_80000001        CI=0    LOOP_H  0
001B  8591 ;11 000 111 010 00 1110 1010 01 11 * 011111 E  R.Q (A.AND.B).SR Q_WX        SHF
001B  8592 ;ALU 80000001 QREG 80000001 PSL<C> 0    RESULTS: WB=40000000, QREG=40000000,
001B  8593 ;                                       RESULTS: DREG=0
001B  8594 ; NO FLAGS ARE SET AND OUTPUT IS DISABLED
001B  8595 ;
001B  8596 ;14
001B  8597 ;DSIZE_1_H 0    DSIZE_0_H 0     ALUSO_H 0
001B  8598 ;AMUX(D)_0      BMUX(R1)_0      CI=1    LOOP_H  0
001B  8599 ;00 000 011 001 10 1001 1010 00 00 * 011110 D D.R2 (A.AND.B).SR SQL.D_WX  1   1
001B  8600 ;ALU 0  QREG 1  PSL<C> 0        RESULTS: QREG=3, DREG=80000000
001B  8601 ;
001B  8602 ;15
001B  8603 ;DSIZE_1_H 0    DSIZE_0_H 0     ALUSO_H 0
001B  8604 ;AMUX(D)_80000000       BMUX(RA)_7FFFFFFF       CI=0    LOOP_H  0
001B  8605 ;00 000 111 011 00 1001 1110 10 10 * 111110 D D.R2 A.AND.NB SQL.WX_D.D_WX  ROT
001B  8606 ;ALU 80000000   QREG 80000001   PSL<C> 0        RESULTS: QREG=3, DREG=80000000
001B  8607 ;
001B  8608 ;16
001B  8609 ;DSIZE_1_H 0    DSIZE_0_H 0     ALUSO_H 0
001B  8610 ;AMUX_0         BMUX(S)_0       CI=1    LOOP_H  0
001B  8611 ;00 001 000 001 10 1101 1011 10 00 * 111111 D 0.S (A.AND.B).SL D_WX     1   1
001B  8612 ;ALU 0          QREG 0          PSL<C> 1        WB=0    QREG 0  DREG 0
001B  8613 ;
001B  8614 ;17
001B  8615 ;DSIZE_1_H 0    DSIZE_0_H 0     ALUSO_H 0
001B  8616 ;AMUX(D)_80000001       BMUX(R5)_0      CI=PSL<C>=1     LOOP_H  0
001B  8617 ;00 001 001 011 11 1001 1011 10 11 * 111110 D   DIVSLOW+
001B  8618 ;ALU 80000001   QREG 0  PSL<C> 1        RESULTS: QREG=1, DREG=2             ROT
001B  8619 ;
001B  8620 ;18
001B  8621 ;DSIZE_1_H 0    DSIZE_0_H 0     ALUSO_H 0
001B  8622 ;AMUX(D)_80000000       BMUX(R4)_80000000       CI=PSL<C>=1     LOOP_H  0
001B  8623 ;00 001 011 001 11 1001 1011 01 10 * 111111 D D.R2 (A.AND.B).SL SQR.D_WX 1   1
001B  8624 ;ALU 80000000   QREG 1  PSL<C> 1        RESULTS: QREG=, DREG=
001B  8625 ;
001B  8626 ; ASSUMPTIONS
001B  8627 ;
001B  8628 ;       This test requires no signals from any other CPU modules but assumes an
```

```
                        8629 ;       operational RBUS, MBUS, WBUS, SBUS, CCC Gate array and CLA Gate array.
                        8630 ;
                        8631 ; ERROR DESCRIPTION
                        8632 ;
                        8633 ; SUBTEST 1 Data Expected    SUBTEST 2 Data Expected    SUBTEST 3 Data Expected
001B 8634 ;1                    0                          0                          0
001B 8635 ;2                    3                          3                          3
001B 8636 ;3                    40000000                   0                          0
001B 8637 ;4                    3                          0                          3
001B 8638 ;5                    80000001                   40000000                   0
001B 8639 ;6                    2                          C0000000                   2
001B 8640 ;7                    0                          0                          0
001B 8641 ;8                    C0000000                   1                          C0000000
001B 8642 ;9                    40000000                   40000000                   40000000
001B 8643 ;10                   40000000                   40000000                   40000000
001B 8644 ;11                   1                          2                          1
001B 8645 ;12                   0                          80000000                   0
001B 8646 ;13                   40000000                   40000000                   0
001B 8647 ;14                                              3                          80000000
001B 8648 ;15                                              3                          80000000
001B 8649 ;16                                              0                          0
001B 8650 ;17                                              1                          0
001B 8651 ;18                                              0                          0
001B 8652 ;
001B 8653 ; Data Received: SUBTEST 1              SUBTEST 2              SUBTEST 3
001B 8654 ;
001B 8655 ;--
001B 8656 ;*****************************************************************************
001B 8657
001B 8658         SUBTEST 1
001E
001E        ;/////////////////////////////////////////////////////////////////////////////
001E 8659
001E 8660         INITIALIZE                          ;DCLO low,Init scratchpads
0020 8661         LOOP    I,1,13                      ;Set loop count
0027 8662         LOADREG WDREG,8$,I                  ;Load RDM D Register for PSL 'C' bit
002F 8663         LDFIELD 13$,I,Byte,2$,34,39         ;Load RSRC Field in microword-loads D
0040 8664         LDFIELD 12$,I,Byte,2$,64,68         ;Load MSRC Field in microword-loads Q
0051 8665         LDFIELD 7$,I,Byte,3$,34,39          ;Load RSRC Field in test microword
0062 8666         LDFIELD 6$,I,Byte,3$,64,68          ;Load MSRC Field in test microword
0073 8667         LDFIELD 5$,I,Word,3$,48,57          ;Load ALPCTL Field in test microword
0084 8668         LDFIELD 4$,I,Byte,3$,58,63          ;Load ROTATE Field in test microword
0095 8669         LDFIELD 14$,I,Byte,3$,40,41         ;Load DTYPE Field in test microword
00A6 8670         LOADDCS 1$,,0,9                     ;Load the test micro instrs.
00AD 8671         ERRLOOP                             ;Set the error loop
00AF 8672         FETCH   0                           ;Get the first micro word
00B4 8673         BRSTCLK 8                           ;Clock stop location 8
00B8 8674         CMPREG  WHREG,9$,I,Long             ;Check the WBus received data
00C1 8675         IFERROR ,<<ALK>>                    ;Continue if no error else call out ALK
00C7 8676         ENDLOOP I                           ;Continue with loop
00CA 8677
00CA 8678         SUBTEST 2
00CD
00CD        ;/////////////////////////////////////////////////////////////////////////////
00CD 8679
```

```
                        00CD  8680          LOOP    I,1,18                  ;Set loop count
                        00D4  8681          LOADREG WDREG,8$,I              ;Load RDM D Register for PSL 'C' bit
                        00DC  8682          LDFIELD 13$,I,Byte,2$,34,39     ;Load RSRC Field in microword-loads D
                        00ED  8683          LDFIELD 12$,I,Byte,2$,64,68     ;Load MSRC Field in microword-loads Q
                        00FE  8684          LDFIELD 7$,I,Byte,3$,34,39      ;Load RSRC Field in test microword
                        010F  8685          LDFIELD 6$,I,Byte,3$,64,68      ;Load MSRC Field in test microword
                        0120  8686          LDFIELD 5$,I,Word,3$,48,57      ;Load ALPCTL Field in test microword
                        0131  8687          LDFIELD 4$,I,Byte,3$,58,63      ;Load ROTATE Field in test microword
                        0142  8688          LDFIELD 14$,I,Byte,3$,40,41     ;Load DTYPE Field in test microword
                        0153  8689          LOADDCS 1$,,0,9                 ;Load the test micro instrs.
                        015A  8690          LOADDCS 15$,,8,2                ;Overlay outputs
                        0161  8691          ERRLOOP                         ;Set the error loop
                        0163  8692          FETCH   0                       ;Get the first micro word
                        0168  8693          BRSTCLK 9                       ;Clock stop location 9
                        016C  8694          CMPREG  WHREG,10$,I,Long        ;Check the WBus received data
                        0175  8695          IFERROR ,<<ALK>>                ;Continue if no error else call out ALK
                        017B  8696          ENDLOOP I                       ;Continue with loop
                        017E  8697
                        017E  8698          SUBTEST 3
                        0181
                        0181        ;////////////////////////////////////////////////////////////////////////////////
                        0181  8699
                        0181  8700          LOOP    I,1,18                  ;Set loop count
                        0188  8701          LOADREG WDREG,8$,I              ;Load RDM D Register for PSL 'C' bit
                        0190  8702          LDFIELD 13$,I,Byte,2$,34,39     ;Load RSRC Field in microword-loads D
                        01A1  8703          LDFIELD 12$,I,Byte,2$,64,68     ;Load MSRC Field in microword-loads Q
                        01B2  8704          LDFIELD 7$,I,Byte,3$,34,39      ;Load RSRC Field in test microword
                        01C3  8705          LDFIELD 6$,I,Byte,3$,64,68      ;Load MSRC Field in test microword
                        01D4  8706          LDFIELD 5$,I,Word,3$,48,57      ;Load ALPCTL Field in test microword
                        01E5  8707          LDFIELD 4$,I,Byte,3$,58,63      ;Load ROTATE Field in test microword
                        01F6  8708          LDFIELD 14$,I,Byte,3$,40,41     ;Load DTYPE Field in test microword
                        0207  8709          LOADDCS 1$,,0,9                 ;Load the test micro instrs.
                        020E  8710          LOADDCS 16$,,8,3                ;Overlay outputs
                        0215  8711          ERRLOOP                         ;Set the error loop
                        0217  8712          FETCH   0                       ;Get the first micro word
                        021C  8713          BRSTCLK <^xA>                   ;Clock stop location A
                        0220  8714          CMPREG  WHREG,11$,I,Long        ;Check the WBus received data
                        0229  8715          IFERROR ,<<ALK>>                ;Continue if no error else call out ALK
                        022F  8716          ENDLOOP I                       ;Continue with loop
                        0232  8717          SKIP                            ;End of test
                        0239  8718
                        0239  8719 BEGIN_TEST_DATA
                        0239
                        0239        ;-------------------------------------------------------------------------
                        0239  8720 ;+
                        0239  8721 ; The following micro words are used to test the ALK Gate Array.
                        0239  8722 ;-
                        0239  8723
                        0239  8724 1$:
U 00, 7700,C800,0364,0300,0470,1801   0239  8725 ;% 00    NOP                             ;NOP micro instruction
U 01, 7700,3800,3FFF,FFFF,8470,1802   0244  8729 ;% 01    LONLIT_[80000000]
U 02, 7700,0868,0247,03D4,0470,1803   024F  8733 ;% 02    M[TEMP8]_LONLIT                 ;SPW/MLONG
U 03, 7700,7800,3FFF,FFFE,8470,1804   025A  8737 ;% 03    LONLIT_[80000002]
U 04, 7700,4869,0247,03D4,0470,1805   0265  8741 ;% 04    M[TEMP9]_LONLIT                 ;SPW/MLONG
                        0270  8745 ;
```

```
                                0270  8746 ; Load Q register and D register with conditioning data RSRC/13$, MSRC/12$
                                0270  8747 ;
                                0270  8748 2$:
U 05, 7700,4800,0057,0300,0470,1806  0270  8749 ;% 05   ALPCTL/WX_D_R.Q_M,RSRC/TEMP0,MSRC/TEMP0
                                027B  8753 ;
                                027B  8754 ; Set C bit in PSL
                                027B  8755 ;
U 06, 5700,4800,0364,0300,0070,1807  027B  8756 ;% 06   WB_RDM,CCPSL/PSL_WB.CCBR_ALUS,MUX/Z.S,ALU/OR,DQ1/NOP
                                0286  8760 ;
                                0286  8761 ; ROT/4$,ALPCTL/5$,MSRC/6$,RSRC/7$,DTYPE/14$ = test micro word
                                0286  8762 ;
                                0286  8763 3$:
U 07, 6700,8800,5B64,0300,0470,1808  0286  8764 ;% 07   RDM_WB,ROT/ZERO,ALPCTL/NOP,MSRC/TEMP0,RSRC/TEMP0
U 08, 7300,4800,0364,0300,0470,1809  0291  8768 ;% 08   STOP.CLOCK
                                029C  8772
                                029C  8773 15$:
U 08, 6700,0800,02C7,0300,0470,1809  029C  8774 ;% 08   RDM_WB,ALPCTL/WX_Q.Q_D          ;Read Q Register into RDM
U 09, 7300,4800,0364,0300,0470,180A  02A7  8778 ;% 09   STOP.CLOCK                     ;Stop clock and check results
                                02B2  8782
                                02B2  8783 16$:
U 08, 6700,0800,02C7,0300,0470,1809  02B2  8784 ;% 08   RDM_WB,ALPCTL/WX_Q.Q_D          ;Load D Register into Q Register
U 09, 6700,0800,02C7,0300,0470,180A  02BD  8788 ;% 09   RDM_WB,ALPCTL/WX_Q.Q_D          ;Read Q(D) Register into RDM
U 0A, 7300,C800,0364,0300,0470,180B  02C8  8792 ;% 0A   STOP.CLOCK                     ;Stop clock and check results
                                02D3  8796
                             0A 02D3  8797 4$:      .Byte ^x0A                       ;TABLE 1 ROTATE Field data
                             0F 02D4  8798          .Byte ^x0F
                             0F 02D5  8799          .Byte ^x0F
                             2B 02D6  8800          .Byte ^x2B
                             0C 02D7  8801          .Byte ^x0C
                             10 02D8  8802          .Byte ^x10
                             16 02D9  8803          .Byte ^x16
                             0F 02DA  8804          .Byte ^x0F
                             09 02DB  8805          .Byte ^x09
                             09 02DC  8806          .Byte ^x09
                             2C 02DD  8807          .Byte ^x2C
                             0C 02DE  8808          .Byte ^x0C
                             28 02DF  8809          .Byte ^x28
                             26 02E0  8810          .Byte ^x26
                             2C 02E1  8811          .Byte ^x2C
                             06 02E2  8812          .Byte ^x06
                             2F 02E3  8813          .Byte ^x2F
                             27 02E4  8814          .Byte ^x27
                                02E5  8815
                           004F 02E5  8816 5$:      .Word ^x04F              ;ALPCTL Field Data
                           015F 02E7  8817          .Word ^x15F
                           0108 02E9  8818          .Word ^x108
                           006F 02EB  8819          .Word ^x06F
                           00D5 02ED  8820          .Word ^x0D5
                           024F 02EF  8821          .Word ^x24F
                           01C9 02F1  8822          .Word ^x1C9
                           021A 02F3  8823          .Word ^x21A
                           02EB 02F5  8824          .Word ^x2EB
                           02EB 02F7  8825          .Word ^x2EB
                           00EE 02F9  8826          .Word ^x0EE
                           006D 02FB  8827          .Word ^x06D
```

```
                03A9  02FD  8828          .Word ^x3A9
                0268  02FF  8829          .Word ^x268
                027A  0301  8830          .Word ^x27A
                036E  0303  8831          .Word ^x36E
                026E  0305  8832          .Word ^x26E
                026D  0307  8833          .Word ^x26D
                      0309  8834
                  05  0309  8835 6$:      .Byte   5                       ;TABLE 3 MSRC Field data
                  01  030A  8836          .Byte   1
                  05  030B  8837          .Byte   5
                  05  030C  8838          .Byte   5
                  02  030D  8839          .Byte   2
                  01  030E  8840          .Byte   1
                  05  030F  8841          .Byte   5
                  01  0310  8842          .Byte   1
                  01  0311  8843          .Byte   1
                  01  0312  8844          .Byte   1
                  01  0313  8845          .Byte   1
                  04  0314  8846          .Byte   4
                  01  0315  8847          .Byte   1
                  01  0316  8848          .Byte   1
                  01  0317  8849          .Byte   1
                  01  0318  8850          .Byte   1
                  01  0319  8851          .Byte   1
                  01  031A  8852          .Byte   1
                      031B  8853
                  01  031B  8854 7$:      .Byte   1                       ;TABLE 4 RSRC Field data
                  05  031C  8855          .Byte   5
                  01  031D  8856          .Byte   1
                  05  031E  8857          .Byte   5
                  01  031F  8858          .Byte   1
                  02  0320  8859          .Byte   2
                  01  0321  8860          .Byte   1
                  01  0322  8861          .Byte   1
                  01  0323  8862          .Byte   1
                  01  0324  8863          .Byte   1
                  01  0325  8864          .Byte   1
                  04  0326  8865          .Byte   4
                  05  0327  8866          .Byte   5
                  01  0328  8867          .Byte   1
                  0A  0329  8868          .Byte ^xA
                  01  032A  8869          .Byte   1
                  05  032B  8870          .Byte   5
                  04  032C  8871          .Byte   4
                      032D  8872
                  00  032D  8873 8$:      .Byte   0                       ;TABLE 5 PSL 'C' Bit data-to be loaded
                  00  032E  8874          .Byte   0
                  00  032F  8875          .Byte   0
                  00  0330  8876          .Byte   0
                  00  0331  8877          .Byte   0
                  01  0332  8878          .Byte   1
                  01  0333  8879          .Byte   1
                  01  0334  8880          .Byte   1
                  00  0335  8881          .Byte   0
                  00  0336  8882          .Byte   0
```

```
                    01  0337  8883          .Byte   1
                    00  0338  8884          .Byte   0
                    00  0339  8885          .Byte   0
                    00  033A  8886          .Byte   0
                    00  033B  8887          .Byte   0
                    01  033C  8888          .Byte   1
                    01  033D  8889          .Byte   1
                    01  033E  8890          .Byte   1
                        033F  8891
00000000            033F  8892 9$:      .Long   0                       ;TABLE 6 WBus data - to be checked
00000003            0343  8893          .Long   3
40000000            0347  8894          .Long ^x40000000
00000003            034B  8895          .Long   3
80000001            034F  8896          .Long ^x80000001
00000002            0353  8897          .Long   2
00000000            0357  8898          .Long   0
C0000000            035B  8899          .Long ^xC0000000
40000000            035F  8900          .Long ^x40000000
40000000            0363  8901          .Long ^x40000000
00000001            0367  8902          .Long   1
00000000            036B  8903          .Long   0
40000000            036F  8904          .Long ^x40000000
                    0373  8905
00000000            0373  8906 10$:     .Long   0                       ;TABLE 7 Q Register data-to be checked
00000003            0377  8907          .Long   3
00000000            037B  8908          .Long   0
00000000            037F  8909          .Long   0
40000000            0383  8910          .Long ^x40000000
C0000000            0387  8911          .Long ^xC0000000
00000000            038B  8912          .Long   0
00000001            038F  8913          .Long   1
40000000            0393  8914          .Long ^x40000000
40000000            0397  8915          .Long ^x40000000
00000002            039B  8916          .Long   2
80000000            039F  8917          .Long ^x80000000
40000000            03A3  8918          .Long ^x40000000
00000003            03A7  8919          .Long   3
00000003            03AB  8920          .Long   3
00000000            03AF  8921          .Long   0
00000001            03B3  8922          .Long   1
00000000            03B7  8923          .Long   0
                    03BB  8924
00000000            03BB  8925 11$:     .Long   0                       ;TABLE 8 D Register data-to be checked
00000003            03BF  8926          .Long   3
00000000            03C3  8927          .Long   0
00000003            03C7  8928          .Long   3
00000000            03CB  8929          .Long   0
00000002            03CF  8930          .Long   2
00000000            03D3  8931          .Long   0
C0000000            03D7  8932          .Long ^xC0000000
40000000            03DB  8933          .Long ^x40000000
40000000            03DF  8934          .Long ^x40000000
00000001            03E3  8935          .Long   1
00000000            03E7  8936          .Long   0
00000000            03EB  8937          .Long   0
```

```
                    80000000  03EF  8938          .Long ^x80000000
                    80000000  03F3  8939          .Long ^x80000000
                    00000000  03F7  8940          .Long   0
                    00000000  03FB  8941          .Long   0
                    00000000  03FF  8942          .Long   0
                              0403  8943
                          01  0403  8944 12$:      .Byte   1               ;Q Register data to be loaded - MSRC Field
                          01  0404  8945          .Byte   1
                          01  0405  8946          .Byte   1
                          02  0406  8947          .Byte   2
                          04  0407  8948          .Byte   4
                          05  0408  8949          .Byte   5
                          05  0409  8950          .Byte   5
                          02  040A  8951          .Byte   2
                          05  040B  8952          .Byte   5
                          05  040C  8953          .Byte   5
                          05  040D  8954          .Byte   5
                          01  040E  8955          .Byte   1
                          05  040F  8956          .Byte   5
                          02  0410  8957          .Byte   2
                          05  0411  8958          .Byte   5
                          01  0412  8959          .Byte   1
                          01  0413  8960          .Byte   1
                          02  0414  8961          .Byte   2
                              0415  8962
                          01  0415  8963 13$:      .Byte   1               ;D Register data to be loaded - RSRC Field
                          01  0416  8964          .Byte   1
                          01  0417  8965          .Byte   1
                          01  0418  8966          .Byte   1
                          01  0419  8967          .Byte   1
                          05  041A  8968          .Byte   5
                          01  041B  8969          .Byte   1
                          04  041C  8970          .Byte   4
                          04  041D  8971          .Byte   4
                          04  041E  8972          .Byte   4
                          01  041F  8973          .Byte   1
                          01  0420  8974          .Byte   1
                          01  0421  8975          .Byte   1
                          01  0422  8976          .Byte   1
                          04  0423  8977          .Byte   4
                          01  0424  8978          .Byte   1
                          05  0425  8979          .Byte   5
                          04  0426  8980          .Byte   4
                              0427  8981
                          00  0427  8982 14$:      .Byte   0               ;DTYPE Data to be Loaded
                          00  0428  8983          .Byte   0
                          00  0429  8984          .Byte   0
                          00  042A  8985          .Byte   0
                          00  042B  8986          .Byte   0
                          00  042C  8987          .Byte   0
                          00  042D  8988          .Byte   0
                          00  042E  8989          .Byte   0
                          01  042F  8990          .Byte   1
                          01  0430  8991          .Byte   1
                          02  0431  8992          .Byte   2
```

```
                     03  0432  8993          .Byte   3
                     03  0433  8994          .Byte   3
                     00  0434  8995          .Byte   0
                     00  0435  8996          .Byte   0
                     00  0436  8997          .Byte   0
                     00  0437  8998          .Byte   0
                     00  0438  8999          .Byte   0
                         0439  9000
                         0439  9001 BEGIN_CPU_DATA
                         0002  9002
                         0002  9003 RTEMP_DATA
                         0001  9004
                00000000 0001  9005          .Long     0             ;TEMP0 - IGNORE
                00000000 0005  9006          .Long     0             ;TEMP1
                00000001 0009  9007          .Long     1             ;TEMP2
                00000003 000D  9008          .Long     3             ;TEMP3
                80000000 0011  9009          .Long   ^x80000000      ;TEMP4
                80000001 0015  9010          .Long   ^x80000001      ;TEMP5
                FFFFFFFF 0019  9011          .Long   ^xFFFFFFFF      ;TEMP6
                FFFFFFFE 001D  9012          .Long   ^xFFFFFFFE      ;TEMP7
                00000000 0021  9013          .Long     0             ;RTEMP8
                00000000 0025  9014          .Long     0             ;RTEMP9
                7FFFFFFF 0029  9015          .Long   ^x7FFFFFFF      ;RTEMPA
                7FFFFFFE 002D  9016          .Long   ^x7FFFFFFE      ;RTEMPB
                         0031  9017
                         0031  9018 END_CPU_DATA
                         0439  9019
                         0439  9020 END_TEST_DATA
                         0439
                         0439       ;-----------------------------------------------------------------------------
                         0439  9021
                         0439  9022 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
001B        .SBTTL  "TEST   02F     ALK Gate Array Test #4
001B  9024 ;*******************************************************************************
001B  9025 ;++
001B  9026 ;
001B  9027 ; FUNCTIONAL DESCRIPTION
001B  9028 ;
001B  9029 ;       This test, which consists of three subtests, continues testing of the
001B  9030 ;       ALU shift and Q Register shift functions of the ALK Gate Array Chip.
001B  9031 ;       This test has the ALUSO Flag set.
001B  9032 ;
001B  9033 ; TEST ALGORITHM
001B  9034 ;
001B  9035 ;       SUBTEST 1
001B  9036 ;
001B  9037 ;       1. Load the micro words to perform the test.
001B  9038 ;       2. Load appropriate micro fields.
001B  9039 ;       3. Execute the micro sequence.
001B  9040 ;               1. Condition the DPM with data from Tables 3 thru 7.
001B  9041 ;               2. Test the DPM with data from Tables 1, 2 and 8.
001B  9042 ;               3. Read WBUS into RDM.
001B  9043 ;       4. Check the results of the WBUS read with TABLE 9.
001B  9044 ;
001B  9045 ;       SUBTEST 2
001B  9046 ;
001B  9047 ;       1. Load the micro words to perform the test.
001B  9048 ;       2. Load appropriate micro fields.
001B  9049 ;       3. Execute the micro sequence.
001B  9050 ;               1. Condition the DPM with data from Tables 3 thru 7.
001B  9051 ;               2. Test the DPM with data from Tables 1, 2 and 8.
001B  9052 ;               3. Read WBUS into RDM.
001B  9053 ;       4. Check the results of the WBUS read (Q Register) with TABLE 10.
001B  9054 ;
001B  9055 ;       SUBTEST 3
001B  9056 ;
001B  9057 ;       1. Load the micro words to perform the test.
001B  9058 ;       2. Load appropriate micro fields.
001B  9059 ;       3. Execute the micro sequence.
001B  9060 ;               1. Condition the DPM with data from Tables 3 thru 7.
001B  9061 ;               2. Test the DPM with data from Tables 1, 2 and 8.
001B  9062 ;               3. Read WBUS into RDM.
001B  9063 ;       4. Check the results of the WBUS read (D Register) with TABLE 11.
001B  9064 ;
001B  9065 ; DATA PATTERNS Including the initial load the D and Q Registers
001B  9066 ;
001B  9067 ;       AMUX            BMUX            QREG            DREG
001B  9068 ;
001B  9069 ;1      FFFFFFFF        1               1               FFFFFFFF
001B  9070 ;2      80000001        80000001        80000000        80000001
001B  9071 ;3      0               80000000        80000000        0
001B  9072 ;4      80000000        7FFFFFFF        80000001        80000000
001B  9073 ;
001B  9074 ;
001B  9075 ; LOGIC DESCRIPTION
001B  9076 ;
001B  9077 ;This test checks the ALK Gate Array's gates for proper operation. These
```

```
001B ;patterns were developed using the XLOG Simulator.
001B ;
001B ;The following display is the simulator output,with additions to make it more
001B ;legible. The AMUX and BMUX represent the two inputs to the ALU. The PSLC_H
001B ;input represents the PSL 'C' bit, QMSB_H represents the Most Significant Bit
001B ;of the Q Register while QR00_H represents the LSB of the Q Register. ALU31_H
001B ;represents the MSB of the ALU and ALU00_H equals the LSB of the ALU. ROT_5_H
001B ;thru ROT_0_H are the ROTATE Field of the Micro Word. OP_9_H thru OP_0_H are
001B ;the ALPCTL Field of the Micro Word. WMUX_CONT indicates whether the WMUX
001B ;Tri-State Drivers are active for any given test. MUX, ALU and DQ CONT is the
001B ;decode of the simulator output recognizable by the Assembler. The two columns
001B ;under the ROTATE heading decode the ROTATE Field output according to the DPM
001B ;Functional Specification. The line showing ALU and some data and QREG and some
001B ;data represent the necessary data in both entities to promote a successful and
001B ;meaningful test for that particular OPCODE and ROTATE Field.
001B ;
001B ; ALUSO FLAG IS SET AND OUTPUT IS ENABLED
001B ;
001B ;DD ALP QQR RRR RR OOOO OOOO OO AA * AAQQQQ W
001B ;SS LOS MRO OOO OO PPPP PPPP PP LL * SSSSSS M
001B ;II UOL SOT TTT TT |||| |||| || UU * IIIIII U
001B ;ZZ SPC B0| ||| || 9876 5432 10 30 * OOOOOO X
001B ;EE O|| ||5 432 10 |||| |||| || 10 * ||||||
001B ;|| |HH HH| ||| || HHHH HHHH HH || * 303100 C
001B ;10 H     H HHH HH              HH * 101570 O
001B ;||                                * |||||| N                          ROTATE
001B ;HH                                * LLLLLL T MUX   ALU     DQCON       ASI/QSI
001B ;*****************************************************************************
001B ;1
001B ;DSIZE_1_H 0    DSIZE_0_H 0     ALUSO_H 1
001B ;AMUX(D)_FFFFFFFF       BMUX(Q)_1       CI=PSL<C>=1     LOOP_H  0
001B ;00 101 011 010 11 1011 0110 11 01 * 110111 E D.Q2 A+B+CI.SR   SQR.D_WX     SHF
001B ;ALU 1  QREG 1  PSL<C> 1        RESULTS: WB=0, QREG=80000000, DREG=0
001B ;
001B ;2
001B ;DSIZE_1_H 0    DSIZE_0_H 1     ALUSO_H 1
001B ;AMUX(D)_80000001       BMUX(Q)_80000001        CI=ALKC=0       LOOP_H  0
001B ;01 100 101 001 01 1011 1010 10 11 * 011110 E D.Q2    AND.SR    SQL.D_WX  1  1
001B ;ALU 80000001 QREG 80000000  PSL<C> 0 RESULTS: WB=C0000000,QREG=1,DREG=C0000000
001B ;
001B ; ALUSO FLAG IS SET AND OUTPUT IS DISABLED
001B ;
001B ;3
001B ;DSIZE_1_H 0    DSIZE_0_H 0     ALUSO_H  1
001B ;AMUX(D)_0      BMUX(R3)_80000000       CI=PSL<C>=1     LOOP_H  0
001B ;00 101 101 101 11 1001 1111 10 10 * 101110 D   DIVSLOW-                1   0
001B ;ALU 80000000   QREG 80000000   PSL<C> 1        RESULTS: QREG=1, DREG=0
001B ;
001B ;4
001B ;DSIZE_1_H 0    DSIZE_0_H 1     ALUSO_H 1
001B ;AMUX(D)_80000000       BMUX(R5)_7FFFFFFF       CI=PSL<C>=1     LOOP_H  0
001B ;01 101 110 111 11 1001 1110 11 10 * 111011 D   MULSLOW+                PSLC
001B ;ALU 80000000   QREG 80000001   PSL<C> 1        RESULTS: QREG=40008000, DREG=0
001B ;
001B ; ASSUMPTIONS
```

```
                        001B  9133 ;
                        001B  9134 ;       This test requires no signals from any other CPU modules but assumes an
                        001B  9135 ;       operational RBUS, MBUS, WBUS, SBUS, CCC Gate array and CLA Gate array.
                        001B  9136 ;
                        001B  9137 ; ERROR DESCRIPTION
                        001B  9138 ;
                        001B  9139 ; SUBTEST 1 Data Expected    SUBTEST 2 Data Expected    SUBTEST 3 Data Expected
                        001B  9140 ;1               0                    80000000                   0
                        001B  9141 ;2               C0000000             1                          C0000000
                        001B  9142 ;3                 -                  0                          3
                        001B  9143 ;4                 -                  40008000                   0
                        001B  9144 ;
                        001B  9145 ; Data Received: SUBTEST 1             SUBTEST 2                  SUBTEST 3
                        001B  9146 ;
                        001B  9147 ;--
                        001B  9148 ;*******************************************************************************
                        001B  9149
                        001B  9150         SUBTEST 1
                        001E
                        001E           ;////////////////////////////////////////////////////////////////////////////
                        001E  9151
                        001E  9152         INITIALIZE                          ;DCLO low,Init scratchpads
                        0020  9153         LOOP    I,1,2                       ;Set loop count
                        0027  9154         LOADREG WDREG,8$,I                  ;Load RDM D Register for PSL 'C' bit
                        002F  9155         LDFIELD 13$,I,Byte,2$,34,39         ;Load RSRC Field in microword-loads D
                        0040  9156         LDFIELD 12$,I,Byte,2$,64,68         ;Load MSRC Field in microword-loads Q
                        0051  9157         LDFIELD 7$,I,Byte,3$,34,39          ;Load RSRC Field in test microword
                        0062  9158         LDFIELD 14$,I,Byte,3$,40,41         ;Load DTYPE Field in test microword
                        0073  9159         LDFIELD 6$,I,Byte,3$,64,68          ;Load MSRC Field in test microword
                        0084  9160         LDFIELD 4$,I,Byte,3$,58,63          ;Load ROTATE Field in test microword
                        0095  9161         LDFIELD 5$,I,Word,3$,48,57          ;Load ALPCTL Field in test microword
                        00A6  9162         LOADDCS 1$,,0,6                     ;Load the test micro instrs.
                        00AD  9163         ERRLOOP                             ;Set the error loop
                        00AF  9164         FETCH   0                           ;Get the first micro word
                        00B4  9165         BRSTCLK 5                           ;Clock stop location 5
                        00B8  9166         CMPREG  WHREG,9$,I,Long             ;Check the Wbus received data
                        00C1  9167         IFERROR ,<<ALK>>                    ;Continue if no error else call out ALK
                        00C7  9168         ENDLOOP I                           ;Continue with loop
                        00CA  9169
                        00CA  9170         SUBTEST 2
                        00CD
                        00CD           ;////////////////////////////////////////////////////////////////////////////
                        00CD  9171
                        00CD  9172         LOOP    I,1,4                       ;Set loop count
                        00D4  9173         LOADREG WDREG,8$,I                  ;Load RDM D Register for PSL 'C' bit
                        00DC  9174         LDFIELD 13$,I,Byte,2$,34,39         ;Load RSRC Field in microword-loads D
                        00ED  9175         LDFIELD 12$,I,Byte,2$,64,68         ;Load MSRC Field in microword-loads Q
                        00FE  9176         LDFIELD 7$,I,Byte,3$,34,39          ;Load RSRC Field in test microword
                        010F  9177         LDFIELD 14$,I,Byte,3$,40,41         ;Load DTYPE Field in test microword
                        0120  9178         LDFIELD 6$,I,Byte,3$,64,68          ;Load MSRC Field in test microword
                        0131  9179         LDFIELD 4$,I,Byte,3$,58,63          ;Load ROTATE Field in test microword
                        0142  9180         LDFIELD 5$,I,Word,3$,48,57          ;Load ALPCTL Field in test microword
                        0153  9181         LOADDCS 1$,,0,6                     ;Load the test micro instrs
                        015A  9182         LOADDCS 15$,,5,2                    ;Overlay outputs
                        0161  9183         ERRLOOP                             ;Set the error loop
```

```
                        0163  9184          FETCH   0                       ;Get the first micro word
                        0168  9185          BRSTCLK 6                       ;Clock stop location 6
                        016C  9186          CMPREG  WHREG,10$,I,Long        ;Check the WBus received data(Q Reg)
                        0175  9187          IFERROR ,<<ALK>>                ;Continue if no error else call out ALK
                        017B  9188          ENDLOOP I                       ;Continue with loop
                        017E  9189
                        017E  9190          SUBTEST 3
                        0181
                        0181          ;////////////////////////////////////////////////////////////////////////////////
                        0181  9191
                        0181  9192          LOOP    I,1,4                   ;Set loop count
                        0188  9193          LOADREG WDREG,8$,I              ;Load RDM D Register for PSL 'C' bit
                        0190  9194          LDFIELD 13$,I,Byte,2$,34,39     ;Load RSRC Field in microword-loads D
                        01A1  9195          LDFIELD 12$,I,Byte,2$,64,68     ;Load MSRC Field in microword-loads Q
                        01B2  9196          LDFIELD 7$,I,Byte,3$,34,39      ;Load RSRC Field in test microword
                        01C3  9197          LDFIELD 14$,I,Byte,3$,40,41     ;Load DTYPE Field in test microword
                        01D4  9198          LDFIELD 6$,I,Byte,3$,64,68      ;Load MSRC Field in test microword
                        01E5  9199          LDFIELD 4$,I,Byte,3$,58,63      ;Load ROTATE Field in test microword
                        01F6  9200          LDFIELD 5$,I,Word,3$,48,57      ;Load ALPCTL Field in test microword
                        0207  9201          LOADDCS 1$,,0,6                 ;Load the test micro instrs.
                        020E  9202          LOADDCS 16$,,5,3                ;Overlay outputs
                        0215  9203          ERRLOOP                         ;Set the error loop
                        0217  9204          FETCH   0                       ;Get the first micro word
                        021C  9205          BRSTCLK 7                       ;Clock stop location 7
                        0220  9206          CMPREG  WHREG,11$,I,Long        ;Check the WBus received data(D Reg)
                        0229  9207          IFERROR ,<<ALK>>                ;Continue if no error else call out ALK
                        022F  9208          ENDLOOP I                       ;Continue with loop
                        0232  9209          SKIP                            ;End of test
                        0239  9210
                        0239  9211 BEGIN_TEST_DATA
                        0239
                        0239          ;-------------------------------------------------------------------------------
                        0239  9212 ;+
                        0239  9213 ; The following micro words are used to test the ALK Gate Array.
                        0239  9214 ;-
                        0239  9215
                        0239  9216 ;
                        0239  9217 ; Set PSL C bit (Uses LOADREG pseudo-op)
                        0239  9218 ;
                        0239  9219 1$:
U 00, 7700,C800,0364,0300,0470,1801   0239  9220 ;% 00    NOP
U 01, 5700,4800,0364,0300,0A70,1802   0244  9224 ;% 01    WB_RDM,CCPSL/CC_WB.CCBR_ALUS
                        024F  9228 ;
                        024F  9229 ; Load Q register and D register with conditioning data RSRC/13$, MSRC/12$
                        024F  9230 ; then set ALUSO flag
                        024F  9231 ;
                        024F  9232 2$:
U 02, 7700,4800,0057,0300,0470,1803   024F  9233 ;% 02    ALPCTL/WX_D_R.Q_M,RSRC/TEMP0,MSRC/TEMP0
U 03, 7700,8801,0028,0304,0470,1804   025A  9237 ;% 03    ALU/AND.SR,MUX/M.R1,DQ1/NOP,RSRC/TEMP1,MSRC/TEMP1
                        0265  9241 ;
                        0265  9242 ; ROT/4$,ALPCTL/5$,MSRC/6$,RSRC/7$,DTYPE/14$ = test micro word
                        0265  9243 ;
                        0265  9244 3$:
U 04, 6700,4800,5B64,0200,0470,1805   0265  9245 ;% 04    RDM_WB,ROT/ZERO,ALPCTL/NOP,MSRC/TEMP0,RSRC/TEMP0,DTYPE/LONG
U 05, 7300,4800,0364,0300,0470,1806   0270  9249 ;% 05    NOP,STOP.CLOCK
```

```
                                027B  9253 15$:
U 05. 6700,0800,02C7,0300,0470,1806  027B  9254 ;% 05   RDM_WB,ALPCTL/WX_Q.Q_D          ;Read Q Register into RDM
U 06. 7300,C800,0364,0300,0470,1807  0286  9258 ;% 06   NOP,STOP.CLOCK                 ;Stop clock and check results
                                0291  9262 16$:
U 05. 7700,0800,02C7,0300,0470,1806  0291  9263 ;% 05   ALPCTL/WX_Q.Q_D               ;Load D Register into Q Register
U 06. 6700,8800,02C7,0300,0470,1807  029C  9267 ;% 06   RDM_WB,ALPCTL/WX_Q.Q_D        ;Read Q(D) Register into RDM
U 07. 7300,C800,0364,0300,0470,1808  02A7  9271 ;% 07   NOP,STOP.CLOCK                 ;Stop clock and check results
                                02B2  9275
                             2B 02B2  9276 4$:      .Byte ^x2B                       ;TABLE 1 ROTATE Field data
                             25 02B3  9277          .Byte ^x25
                             37 02B4  9278          .Byte ^x37
                             1F 02B5  9279          .Byte ^x1F
                                02B6  9280
                           02DB 02B6  9281 5$:      .Word ^x2DB                      ;TABLE 2 ALPCTL Field Data
                           02EA 02B8  9282          .Word ^x2EA
                           027E 02BA  9283          .Word ^x27E
                           027B 02BC  9284          .Word ^x27B
                                02BE  9285
                             01 02BE  9286 6$:      .Byte   1                        ;TABLE 3 MSRC Field data
                             01 02BF  9287          .Byte   1
                             01 02C0  9288          .Byte   1
                             01 02C1  9289          .Byte   1
                                02C2  9290
                             01 02C2  9291 7$:      .Byte   1                        ;TABLE 4 RSRC Field data
                             01 02C3  9292          .Byte   1
                             03 02C4  9293          .Byte   3
                             05 02C5  9294          .Byte   5
                                02C6  9295
                             01 02C6  9296 8$:      .Byte   1                        ;TABLE 5 PSL 'C' Bit data-to be loaded
                             00 02C7  9297          .Byte   0
                             01 02C8  9298          .Byte   1
                             01 02C9  9299          .Byte   1
                                02CA  9300
                       00000000 02CA  9301 9$:      .Long   0                        ;TABLE 9 WBus data - to be checked
                       C0000000 02CE  9302          .Long ^xC0000000
                                02D2  9303
                       80000000 02D2  9304 10$:     .Long ^x80000000                 ;TABLE 10 Q Register data-to be checked
                       00000001 02D6  9305          .Long   1
                       00000000 02DA  9306          .Long   0
                       40008000 02DE  9307          .Long ^x40008000
                                02E2  9308
                       00000000 02E2  9309 11$:     .Long   0                        ;TABLE 11 D Register data-to be checked
                       C0000000 02E6  9310          .Long ^xC0000000
                       00000003 02EA  9311          .Long   3
                       00000000 02EE  9312          .Long   0
                                02F2  9313
                             02 02F2  9314 12$:     .Byte   2                        ;TABLE 6 Q Register data to be loaded -
                             03 02F3  9315          .Byte   3                        ;MSRC Field
                             03 02F4  9316          .Byte   3
                             04 02F5  9317          .Byte   4
                                02F6  9318
                             06 02F6  9319 13$:     .Byte   6                        ;TABLE 7 Register data to be loaded -
                             04 02F7  9320          .Byte   4                        ;RSRC Data
                             01 02F8  9321          .Byte   1
                             03 02F9  9322          .Byte   3
```

```
                    02FA  9323
                 00 02FA  9324 14$:     .Byte   0                       ;TABLE 8 DTYPE Data to be Loaded
                 01 02FB  9325          .Byte   1
                 00 02FC  9326          .Byte   0
                 01 02FD  9327          .Byte   1
                    02FE  9328
                    02FE  9329 BEGIN_CPU_DATA
                    0002  9330
                    0002  9331 RTEMP_DATA
                    0001  9332
           00000000 0001  9333          .Long     0             ;TEMP0 - IGNORE
           00000000 0005  9334          .Long     0             ;TEMP1
           00000001 0009  9335          .Long     1             ;TEMP2
           80000000 000D  9336          .Long   ^x80000000      ;TEMP3
           80000001 0011  9337          .Long   ^x80000001      ;TEMP4
           7FFFFFFF 0015  9338          .Long   ^x7FFFFFFF      ;TEMP5
           FFFFFFFF 0019  9339          .Long   ^xFFFFFFFF      ;TEMP6
                    001D  9340
                    001D  9341 END_CPU_DATA
                    02FE  9342
                    02FE  9343 END_TEST_DATA
                    02FE
                    02FE        ;-----------------------------------------------------------------------
                    02FE  9344
                    02FE  9345 ENDTEST
```

```
                        001B        .SBTTL  "TEST   030     ALK Gate Array Test #5
                        001B  9347 ;*******************************************************************************
                        001B  9348 ;++
                        001B  9349 ;
                        001B  9350 ; FUNCTIONAL DESCRIPTION
                        001B  9351 ;
                        001B  9352 ;       Subtest 1 of this test checks the ALK Gate Array QSIO 7 and 15 Outputs.
                        001B  9353 ;       Subtest 2 checks the Double Enable Function of the ALK and SAC Gate
                        001B  9354 ;       Arrays.
                        001B  9355 ;
                        001B  9356 ; TEST ALGORITHM
                        001B  9357 ;
                        001B  9358 ;       1. Load the micro words to perform the test.
                        001B  9359 ;       2. Setup a loop of two iterations.
                        001B  9360 ;       3. Select and load DTYPE field of the micro word.
                        001B  9361 ;       4. Execute the micro sequence.
                        001B  9362 ;               1. Load LONLIT Register with Zero.
                        001B  9363 ;               2. Load Q Register from LONLIT Register.
                        001B  9364 ;               3. Execute Test micro word(s).
                        001B  9365 ;               4. Read Q Register into RDM.
                        001B  9366 ;       5. Check the results of the Q Register read.
                        001B  9367 ;
                        001B  9368 ; LOGIC DESCRIPTION
                        001B  9369 ;
                        001B  9370 ;       These tests check the ALK Gate array's QSIO 07 L and QSIO 15 L outputs
                        001B  9371 ;       and the Clock Double Enable Function which is contained within the ALK
                        001B  9372 ;       and SAC Gate Arrays.
                        001B  9373 ;
                        001B  9374 ; ASSUMPTIONS
                        001B  9375 ;
                        001B  9376 ;       This test requires no signals from any other CPU modules but assumes an
                        001B  9377 ;       operational RBUS, MBUS, WBUS, and SAC Gate array.
                        001B  9378 ;
                        001B  9379 ; ERROR DESCRIPTION
                        001B  9380 ;                                               Loop 1   Loop 2
                        001B  9381 ;       Data expected:  SUBTEST 1       00000080 00008000
                        001B  9382 ;                       SUBTEST 2       00000020 00002000
                        001B  9383 ;       Data received:
                        001B  9384 ;--
                        001B  9385 ;*******************************************************************************
                        001B  9386
                        001B  9387         SUBTEST 1
                        001E
                        001E       ;////////////////////////////////////////////////////////////////////////////////
                        001E  9388
                        001E  9389         INITIALIZE                              ;DCLO low,Init scratchpads
                        0020  9390         LOOP    I,1,2                           ;Set loop count
                        0027  9391         LDFIELD 11$,,Byte,2$,48,57              ;Load ALPCTL Field
                        0038  9392         LDFIELD 4$,I,Byte,2$,40,41              ;Load DTYPE Field
                        0049  9393         LOADDCS 1$,,0,6                         ;Load the test micro instrs.
                        0050  9394         ERRLOOP                                 ;Set the error loop
                        0052  9395         FETCH   0                               ;Get the first micro word
                        0057  9396         BRSTCLK 5                               ;Clock stop location 5
                        005B  9397         CMPREG  WHREG,5$,I,Long                 ;Check the received data
                        0064  9398         IFERROR ,<<ALK>,<SAC>,<ALP>>,<<DPM>> ;Continue if no error else call
```

```
                        006D  9399                                    ;out ALK,SAC,ALP Gate Arrays
                        006D  9400          ENDLOOP I                   ;Continue with loop
                        0070  9401
                        0070  9402          SUBTEST 2
                        0073
                        0073        ;////////////////////////////////////////////////////////////////////////////
                        0073  9403
                        0073  9404          LOOP    I,1,2               ;Set loop count
                        007A  9405          LDFIELD 12$,,Byte,3$,48,57  ;Load ALPCTL Field
                        008B  9406          LDFIELD 12$,,Byte,7$,48,57  ;Load ALPCTL Field
                        009C  9407          LDFIELD 4$,I,Byte,3$,40,41  ;Load DTYPE Field
                        00AD  9408          LDFIELD 4$,I,Byte,7$,40,41  ;Load DTYPE Field
                        00BE  9409          LOADDCS 1$,,0,6             ;Load the test micro instrs.
                        00C5  9410          LOADDCS 3$,,3,4             ;Load the test micro instrs.
                        00CC  9411          ERRLOOP                     ;Set the error loop
                        00CE  9412          FETCH   0                   ;Get the first micro word
                        00D3  9413          BRSTCLK 6                   ;Clock stop location 6
                        00D7  9414          CMPREG  WHREG,6$,I,Long     ;Check the received data
                        00E0  9415          IFERROR ,<<ALK>,<SAC>,<ALP>>,<<DPM>> ;Continue if no error else call
                        00E9  9416                                      ;out ALK,SAC,ALP Gate Arrays
                        00E9  9417          ENDLOOP I                   ;Continue with loop
                        00EC  9418          SKIP                        ;End of test
                        00F3  9419
                        00F3  9420 BEGIN_TEST_DATA
                        00F3
                        00F3        ;-------------------------------------------------------------------------
                        00F3  9421 ;+
                        00F3  9422 ; The following micro words are used for ALK Test #5.
                        00F3  9423 ;-
                        00F3  9424
                        00F3  9425 1$:
U 00, 7700,C800,0364,0300,0470,1801  00F3  9426 ;* 00   NOP                         ;NOP Micro instruction
U 01, 7700,7800,7FFF,FFFF,8470,1802  00FE  9430 ;* 01   LONLIT_[0]                 ;Clear LONLIT Register
U 02, 7700,4800,03C7,03D4,0470,1803  0109  9434 ;* 02   Q_LONLIT                   ;Clear Q Register
                                     0114  9438 2$:
U 03, 7700,8800,4364,0000,0470,1804  0114  9439 ;* 03   ALPCTL/NOP,DTYPE/Byte,ROT/GETEXP ;DTYPE/4$,ALPCTL/11$
U 04, 6700,0800,02C7,0300,0470,1805  011F  9443 ;* 04   RDM_Q Q_D                  ;Read Q Register
U 05, 7300,4800,0364,0300,0470,1806  012A  9447 ;* 05   NOP,STOP.CLOCK            ;Stop clock and check results
                                     0135  9451
                                     0135  9452 3$:
U 03, 7700,8800,4364,0000,0470,1804  0135  9453 ;* 03   ALPCTL/NOP,DTYPE/Byte,ROT/GETEXP ;DTYPE/4$,ALPCTL/12$
                                     0140  9457
                                     0140  9458 7$:
U 04, 7700,0800,4364,0000,0470,1805  0140  9459 ;* 04   ALPCTL/NOP,DTYPE/Byte,ROT/GETEXP ;DTYPE/4$,ALPCTL/12$
U 05, 6700,0800,02C7,0300,0470,1806  014B  9463 ;* 05   RDM_Q Q_D                  ;Read Q Register
U 06, 7300,C800,0364,0300,0470,1807  0156  9467 ;* 06   NOP,STOP.CLOCK            ;Stop clock and check results
                                     0161  9471
                              00     0161  9472 4$:      .Byte     0             ;DTYPE Table
                              01     0162  9473          .Byte     1
                                     0163  9474
                          00000080   0163  9475 5$:      .Long     ^x00000080    ;Expected data
                          00008000   0167  9476          .Long     ^x00008000
                                     016B  9477
                          000000E0   016B  9478 6$:      .Long     ^x000000E0    ;    "       "
                          0000E000   016F  9479          .Long     ^x0000E000
```

```
                     0173  9480
027B                 0173  9481 11$:     .Word    ^x27B                    ;MULSLOW+
0279                 0175  9482 12$:     .Word    ^x279                    ;MULFAST+
                     0177  9483
                     0177  9484 END_TEST_DATA
                     0177
                     0177       ;-------------------------------------------------------------------------
                     0177  9485
                     0177  9486 ENDTEST
```

```
001B        .SBTTL  "TEST   031     ALK Gate Array Test #6
001B  9488 ;*******************************************************************************
001B  9489 ;++
001B  9490 ;
001B  9491 ; FUNCTIONAL DESCRIPTION
001B  9492 ;
001B  9493 ;       This test, which consists of three subtests, continues testing the ALU
001B  9494 ;       and Q Register shift functions of the ALK Gate Array Chip.This test has
001B  9495 ;       the ALUSO and LOOP Flags set. The  D  Register output data is not valid
001B  9496 ;       for the last two tests, therefore is not shown.
001B  9497 ;
001B  9498 ; TEST ALGORITHM
001B  9499 ;
001B  9500 ;       SUBTEST 1
001B  9501 ;
001B  9502 ;       1. Load the micro words to perform the test.
001B  9503 ;       2. Setup a loop of four iterations.
001B  9504 ;       3. Load appropriate micro fields.
001B  9505 ;       4. Execute the micro sequence.
001B  9506 ;               1. Condition the DPM with data from Tables 9 and 10.
001B  9507 ;               2. Test the DPM with data from Tables 1 thru 8.
001B  9508 ;               3. Read WBUS into RDM.
001B  9509 ;       5. Check the results of the WBUS read.
001B  9510 ;       6. Looped four times? If not go to step three.
001B  9511 ;
001B  9512 ;       SUBTEST 2
001B  9513 ;
001B  9514 ;       1. Load the micro words to perform the test.
001B  9515 ;       2. Setup a loop of nine iterations.
001B  9516 ;       3. Load appropriate micro fields.
001B  9517 ;       4. Execute the micro sequence.
001B  9518 ;               1. Condition the DPM with data from Tables 9 and 10.
001B  9519 ;               2. Test the DPM with data from Tables 1 thru 8.
001B  9520 ;               3. Read WBUS into RDM.
001B  9521 ;       5. Check the results of the WBUS read (Q Register).
001B  9522 ;       6. Looped nine times? If not go to step three.
001B  9523 ;
001B  9524 ;       SUBTEST 3
001B  9525 ;
001B  9526 ;       1. Load the micro words to perform the test.
001B  9527 ;       2. Setup a loop of nine iterations.
001B  9528 ;       3. Load appropriate micro fields.
001B  9529 ;       4. Execute the micro sequence.
001B  9530 ;               1. Condition the DPM with data from Tables 9 and 10.
001B  9531 ;               2. Test the DPM with data from Tables 1 thru 8.
001B  9532 ;               3. Read WBUS into RDM.
001B  9533 ;       5. Check the results of the WBUS read (D Register).
001B  9534 ;       6. Looped nine times? If not go to step three.
001B  9535 ;
001B  9536 ; DATA PATTERNS Including the initial load the D and Q Registers
001B  9537 ;
001B  9538 ;          AMUX            BMUX            QREG            DREG          RBUS
001B  9539 ;                                                                    For DIVSLOW-
001B  9540 ;                                                                     Instruction
001B  9541 ;
```

```
001B  9542 ;1      80000001        80000001        E0000000        F0000000        F0000000
001B  9543 ;2      80000001        80000001        C0000000        E0000000        E0000000
001B  9544 ;3      80000001        1               C0000000        E0000000        E0000000
001B  9545 ;4      80000001        80000000        C0000000        E0000000        E0000000
001B  9546 ;5      80000001        0               80000000        E0000000        E0000000
001B  9547 ;6      0               0               40              C0000000        C0000000
001B  9548 ;7      0               0               0               C0000000        C0000000
001B  9549 ;8      0               0               80004000        E0000000        E0000000
001B  9550 ;9      0               0               80004000        E0000000        E0000000
001B  9551 ;
001B  9552 ; LOGIC DESCRIPTION
001B  9553 ;
001B  9554 ;This test checks the ALK Gate Array's gates for proper operation. These
001B  9555 ;patterns were developed using the XLOG Simulator.
001B  9556 ;
001B  9557 ;The following display is the simulator output,with additions to make it more
001B  9558 ;legible. The AMUX and BMUX represent the two inputs to the ALU. The PSLC_H
001B  9559 ;input represents the PSL 'C' bit, QMSB_H represents the Most Significant Bit
001B  9560 ;of the Q Register while QR00_H represents the LSB of the Q Register. ALU31_H
001B  9561 ;represents the MSB of the ALU and ALU00_H equals the LSB of the ALU. ROT_5_H
001B  9562 ;thru ROT_0_H are the ROTATE Field of the Micro Word. OP_9_H thru OP_0_H are
001B  9563 ;the ALPCTL Field of the Micro Word. DSIZE_1_H and DSIZE_0_H are the DTYPE
001B  9564 ;Field of the Micro Word. WMUX_CONT indicates whether the WMUX Tri-State
001B  9565 ;Drivers are active for any given test. MUX, ALU and DQ CONT is the decode of
001B  9566 ;the simulator output recognizable by the Assembler. The two columns under the
001B  9567 ;ROTATE heading decode the ROTATE Field output according to the DPM Functional
001B  9568 ;Specification. The line showing ALU and some data and QREG and some data
001B  9569 ;represent the necessary data in both entities to promote a successful and
001B  9570 ;meaningful test for that particular OPCODE, DSIZE and ROTATE Field.
001B  9571 ;
001B  9572 ;
001B  9573 ; LOOP AND ALUSO FLAGS ARE SET AND OUTPUT IS ENABLED
001B  9574 ;
001B  9575 ;                BOOLEAN MAP
001B  9576 ;
001B  9577 ;DD ALP QQR RRR RR 0000 0000 00 AA * AAQQQQ W
001B  9578 ;SS LOS MRO 000 00 PPPP PPPP PP LL * SSSSSS M
001B  9579 ;II UOL S0T TTT TT |||| |||| || UU * IIIIII U
001B  9580 ;ZZ SPC B0| ||| || 9876 5432 10 30 * 000000 X
001B  9581 ;EE O|| ||5 432 10 |||| |||| || 10 * |||||| 
001B  9582 ;|| |HH HH| ||| || HHHH HHHH HH || * 303100 C
001B  9583 ;10 H     H HHH HH                HH * 101570 O
001B  9584 ;||                                 * |||||| N                          ROTATE
001B  9585 ;HH                                 * LLLLLL T MUX   ALU     DQCON      ASI/QSI
001B  9586 ;*******************************************************************************
001B  9587 ;1
001B  9588 ;DSIZE_1_H 0    DSIZE_0_H 1     ALUSO_H 1
001B  9589 ;AMUX(D)_C0000001          BMUX(Q)_C0000001          CI=PSL<C>=0     LOOP_H  1
001B  9590 ;01 110 110 011 11 1011 0100 01 10 * 110111 E D.Q2 A+B+CI       SQR          ROT
001B  9591 ;ALU 80000000 QREG 80000001 PSL<C> 0    RESULTS: WB=80000002, QREG=E0000001,
001B  9592 ;                                       RESULTS: DREG=C0000001
001B  9593 ;
001B  9594 ;2
001B  9595 ;DSIZE_1_H 0    DSIZE_0_H 1     ALUSO_H 1
001B  9596 ;AMUX(D)_80000001          BMUX(Q)_80000001          CI=1    LOOP_H  1
```

```
001B  9597 ;01 110 110 111 10 1011 0000 01 11 * 111111 E D.Q2 A-B-CI       SQR         PSLC
001B  9598 ;ALU 80000001 QREG 80000001 PSL<C> 0    RESULTS: WB=FFFFFFFF, QREG=40000000,
001B  9599 ;                                       RESULTS: DREG=800000001
001B  9600 ;
001B  9601 ;3
001B  9602 ;DSIZE_1_H 1    DSIZE_0_H 1     ALUSO_H 1
001B  9603 ;AMUX(D)_80000001       BMUX(R2)_1      CI=0    LOOP_H  1
001B  9604 ;11 110 111 011 00 1000 1011 01 01 * 111111 E D.R1   AND.SL   Q_WX          ROT
001B  9605 ;ALU 1  QREG 80000001   PSL<C> 0        RESULTS: WB=2, QREG=2, DREG=80000000
001B  9606 ;
001B  9607 ;4
001B  9608 ;DSIZE_1_H 0    DSIZE_0_H 1     ALUSO_H 1
001B  9609 ;AMUX(D)_80000001       BMUX(R3)_80000000       CI=ALKC=0       LOOP_H  1
001B  9610 ;01 111 110 110 01 1000 1010 10 10 * 111111 E D.R1   AND.SR  D_WX          WB30
001B  9611 ;ALU 80000000   QREG 80000001   PSL<C> 1        RESULTS: WB=C0000000,
001B  9612 ;                                       RESULTS: QREG=80000001, DREG=C0000000
001B  9613 ;
001B  9614 ; LOOP AND ALUSO FLAGS ARE SET AND OUTPUT IS DISABLED
001B  9615 ;
001B  9616 ;5
001B  9617 ;DSIZE_1_H 0    DSIZE_0_H 0     ALUSO_H 1
001B  9618 ;AMUX(D)_80000001       BMUX(R1)_0      CI=PSL<C>=0     LOOP_H  1
001B  9619 ;00 110 011 010 11 1001 1011 10 11 * 111110 D D.R2   DIVSLOW+                SHF
001B  9620 ;ALU 80000001   QREG 1  PSL<C> 0        RESULTS: QREG=2, DREG=2
001B  9621 ;
001B  9622 ;6
001B  9623 ;DSIZE_1_H 0    DSIZE_0_H 0     ALUSO_H 1
001B  9624 ;AMUX(D)_0      BMUX(R1)_0      CI=PSL<C>=0     LOOP_H  1
001B  9625 ;00 110 111 110 11 1001 1111 11 00 * 111110 D D.R2      DIVDA           WB30
001B  9626 ;ALU 0  QREG 81 PSL<C> 0        RESULTS: QREG=102, DREG=1
001B  9627 ;
001B  9628 ;7
001B  9629 ;DSIZE_1_H 0    DSIZE_0_H 1     ALUSO_H 1
001B  9630 ;AMUX(D)_0      BMUX(R1)_0      CI=ALKC=0       LOOP_H  1
001B  9631 ;01 110 010 110 01 1001 1111 11 00 * 101110 D D.R2      DIVDA           WB30
001B  9632 ;ALU 0  QREG 1  PSL<C> 0        RESULTS: QREG=2, DREG=1
001B  9633 ;
001B  9634 ;8
001B  9635 ;DSIZE_1_H 0    DSIZE_0_H 1     ALUSO_H 1
001B  9636 ;AMUX(D)_80000001       BMUX(R9)_7FFFFFFF       CI=1    LOOP_H  1
001B  9637 ;01 110 110 111 10 1001 1111 00 00 * 101110 D D.R2      DIVFAST-        PSLC
001B  9638 ;ALU 0  QREG 8001       PSL<C> 0        RESULTS: QREG=20006
001B  9639 ;
001B  9640 ;9
001B  9641 ;DSIZE_1_H 0    DSIZE_0_H 1     ALUSO_H 1
001B  9642 ;AMUX(D)_0      BMUX(R1)_0      CI=1    LOOP_H  1
001B  9643 ;01 111 110 110 10 1001 1010 11 00 * 111111 D D.R2    MULSLOW-        WB30
001B  9644 ;ALU 0  QREG 8001       PSL<C> 1        RESULTS: QREG=4000
001B  9645 ;
001B  9646 ; ASSUMPTIONS
001B  9647 ;
001B  9648 ;       This test requires no signals from any other CPU modules but assumes an
001B  9649 ;       operational RBUS, MBUS, WBUS, SBUS, CCC Gate array and CLA Gate array.
001B  9650 ;
001B  9651 ; ERROR DESCRIPTION
```

```
                        001B  9652 ;
                        001B  9653 ; SUBTEST 1 Data Expected    SUBTEST 2 Data Expected    SUBTEST 3 Data Expected
                        001B  9654 ;
                        001B  9655 ;                    WBUS                     Q REG                    D REG
                        001B  9656 ;
                        001B  9657 ;1                80000002                 E0000000                 C0000001
                        001B  9658 ;2                FFFFFFFF                 40000000                 80000001
                        001B  9659 ;3                2                        2                        80000001
                        001B  9660 ;4                C0000000                 80000001                 C0000000
                        001B  9661 ;5                -                        2                        2
                        001B  9662 ;6                -                        102                      2
                        001B  9663 ;7                -                        2                        3
                        001B  9664 ;8                -                        20006                    -
                        001B  9665 ;9                -                        C000                     -
                        001B  9666 ;
                        001B  9667 ; Data Received: SUBTEST 1                 SUBTEST 2                SUBTEST 3
                        001B  9668 ; LOOP COUNT:
                        001B  9669 ;--
                        001B  9670 ;*******************************************************************************
                        001B  9671
                        001B  9672          SUBTEST 1
                        001E
                        001E               ;////////////////////////////////////////////////////////////////////////
                        001E  9673
                        001E  9674          INITIALIZE                          ;DCLO low,Init scratchpads
                        0020  9675          LOOP     I,1,4                      ;Set loop count
                        0027  9676          LOADREG  WDREG,10$,I                ;Load RDM D Register for PSL 'C' bit
                        002F  9677          LDFIELD  15$,I,Byte,2$,64,68        ;Load MSRC Field in microword-loads Q
                        0040  9678          LDFIELD  16$,I,Byte,2$,34,39        ;Load RSRC Field in microword-loads D
                        0051  9679          LDFIELD  17$,I,Byte,3$,34,39        ;Load RSRC Field for DIVSLOW- instruct.
                        0062  9680          LDFIELD  9$,I,Byte,4$,34,39         ;Load RSRC Field in test microword
                        0073  9681          LDFIELD  11$,I,Byte,4$,40,41        ;Load DTYPE Field in test microword
                        0084  9682          LDFIELD  7$,I,Byte,4$,58,63         ;Load ROTATE Field in test microword
                        0095  9683          LDFIELD  8$,I,Word,4$,48,57         ;Load ALPCTL Field in test microword
                        00A6  9684          LOADDCS  1$,,0,9                    ;Load the test micro instrs.
                        00AD  9685          ERRLOOP                             ;Set the error loop
                        00AF  9686          FETCH    0                          ;Get the first micro word
                        00B4  9687          BRSTCLK  8                          ;Clock stop location 8
                        00B8  9688          CMPREG   WHREG,12$,I,Long           ;Check the WBus received data
                        00C1  9689          IFERROR  ,<<ALK>>                   ;Continue if no error else call out ALK
                        00C7  9690          ENDLOOP  I                          ;Continue with loop
                        00CA  9691
                        00CA  9692          SUBTEST 2
                        00CD
                        00CD               ;////////////////////////////////////////////////////////////////////////
                        00CD  9693
                        00CD  9694          LOOP     I,1,9                      ;Set loop count
                        00D4  9695          LOADREG  WDREG,10$,I                ;Load RDM D Register for PSL 'C' bit
                        00DC  9696          LDFIELD  15$,I,Byte,2$,64,68        ;Load MSRC Field in microword-loads Q
                        00ED  9697          LDFIELD  16$,I,Byte,2$,34,39        ;Load RSRC Field in microword-loads D
                        00FE  9698          LDFIELD  17$,I,Byte,3$,34,39        ;Load RSRC Field for DIVSLOW- instruct.
                        010F  9699          LDFIELD  9$,I,Byte,4$,34,39         ;Load RSRC Field in test microword
                        0120  9700          LDFIELD  11$,I,Byte,4$,40,41        ;Load DTYPE Field in test microword
                        0131  9701          LDFIELD  7$,I,Byte,4$,58,63         ;Load ROTATE Field in test microword
                        0142  9702          LDFIELD  8$,I,Word,4$,48,57         ;Load ALPCTL Field in test microword
```

```
                        0153  9703          LOADDCS 1$.,0,9                 ;Load the test micro instrs.
                        015A  9704          LOADDCS 5$.,8,2                 ;Overlay outputs
                        0161  9705          ERRLOOP                         ;Set the error loop
                        0163  9706          FETCH   0                       ;Get the first micro word
                        0168  9707          BRSTCLK 9                       ;Clock stop location 9
                        016C  9708          CMPREG  WHREG,13$,I,Long        ;Check the WBus received data (Q Reg)
                        0175  9709          IFERROR ,<<ALK>>                ;Continue if no error else call out ALK
                        017B  9710          ENDLOOP I                       ;Continue with loop
                        017E  9711
                        017E  9712          SUBTEST 3
                        0181
                        0181          ;//////////////////////////////////////////////////////////////////////////////
                        0181  9713
                        0181  9714          LOOP    I,1,7                   ;Set loop count
                        0188  9715          LOADREG WDREG,10$,I             ;Load RDM D Register for PSL 'C' bit
                        0190  9716          LDFIELD 15$,I,Byte,2$,64,68     ;Load MSRC Field in microword-loads Q
                        01A1  9717          LDFIELD 16$,I,Byte,2$,34,39     ;Load RSRC Field in microword-loads D
                        01B2  9718          LDFIELD 17$,I,Byte,3$,34,39     ;Load RSRC Field for DIVSLOW- instruct.
                        01C3  9719          LDFIELD 9$,I,Byte,4$,34,39      ;Load RSRC Field in test microword
                        01D4  9720          LDFIELD 11$,I,Byte,4$,40,41     ;Load DTYPE Field in test microword
                        01E5  9721          LDFIELD 7$,I,Byte,4$,58,63      ;Load ROTATE Field in test microword
                        01F6  9722          LDFIELD 8$,I,Word,4$,48,57      ;Load ALPCTL Field in test microword
                        0207  9723          LOADDCS 1$.,0,9                 ;Load the test micro instrs.
                        020E  9724          LOADDCS 6$.,8,3                 ;Overlay outputs
                        0215  9725          ERRLOOP                         ;Set the error loop
                        0217  9726          FETCH   0                       ;Get the first micro word
                        021C  9727          BRSTCLK <^xA>                   ;Clock stop location A
                        0220  9728          CMPREG  WHREG,14$,I,Long        ;Check the WBus received data (D Reg)
                        0229  9729          IFERROR ,<<ALK>>                ;Continue if no error else call out ALK
                        022F  9730          ENDLOOP I                       ;Continue with loop
                        0232  9731          SKIP                            ;End of test
                        0239  9732
                        0239  9733 BEGIN_TEST_DATA
                        0239
                        0239          ;-------------------------------------------------------------------------
                        0239  9734 ;+
                        0239  9735 ; The following micro words are used to test the ALK Gate Array.
                        0239  9736 ;-
                        0239  9737
                        0239  9738 1$:
U 00, 7700,C800,0364,0300,0470,1801     0239  9739 ;% 00    NOP                             ;NOP micro instruction
U 01, 7700,C868,5BE4,0320,0470,1802     0244  9743 ;% 01    M[TEMP8]_R[TEMP8]               ;Load MTEMP8
U 02, 7700,4869,5BE4,0324,0470,1803     024F  9747 ;% 02    M[TEMP9]_R[TEMP9]               ;Load MTEMP9
U 03, 7700,C86A,5BE4,0328,0470,1804     025A  9751 ;% 03    M[TEMP10]_R[TEMP10]             ;Load MTEMP10
                        0265  9755 ;
                        0265  9756 ; Load Q register and D register with conditioning data MSRC/15$, RSRC/16$
                        0265  9757 ; Then set C bit in PSL
                        0265  9758 ;
                        0265  9759 2$:
U 04, 7700,4800,0057,0300,0470,1805     0265  9760 ;% 04    ALPCTL/WX_D_R.Q_M,RSRC/TEMP0,MSRC/TEMP0
U 05, 5700,C800,0364,0300,0070,1806     0270  9764 ;% 05    WB_RDM,CCPSL/PSL_WB.CCBR_ALUS,MUX/Z.S,ALU/OR,DQ1/NOP
                        027B  9768 ;
                        027B  9769 ; Set ALUSO and LOOP flags - RSRC/17$
                        027B  9770 ;
                        027B  9771 3$:
```

```
U 06, 7701,8800,027E,0200,0470,1807  027B  9772 ;% 06   ALPCTL/DIVSLOW-,RSRC/TEMP0,SIZE[LONG]
                                     0286  9776 ;
                                     0286  9777 ; ROT/7$,ALPCTL/8$,RSRC/9$ = test micro word
                                     0286  9778 ;
                                     0286  9779 4$:
U 07, 6700,C801,5B64,0300,0470,1808  0286  9780 ;% 07   RDM_WB,ROT/ZERO,ALPCTL/NOP,MSRC/TEMP1,RSRC/TEMP0
U 08, 7300,4800,0364,0300,0470,1809  0291  9784 ;% 08   NOP,STOP.CLOCK
                                     029C  9788 5$:
U 08, 6700,0800,02C7,0300,0470,1809  029C  9789 ;% 08   RDM_WB,ALPCTL/WX_Q.Q_D          ;Read Q Register into RDM
U 09, 7300,4800,0364,0300,0470,180A  02A7  9793 ;% 09   NOP,STOP.CLOCK                 ;Stop clock and check results
                                     02B2  9797 6$:
U 08, 7700,0800,02C7,0300,0470,1809  02B2  9798 ;% 08   ALPCTL/WX_Q.Q_D               ;Load D Register into Q Register
U 09, 6700,0800,02C7,0300,0470,180A  02BD  9802 ;% 09   RDM_WB,ALPCTL/WX_Q.Q_D        ;Read Q(D) Register into RDM
U 0A, 7300,C800,0364,0300,0470,180B  02C8  9806 ;% 0A   NOP,STOP.CLOCK                 ;Stop clock and check results
                                     02D3  9810
                                0F   02D3  9811 7$:     .Byte ^x0F                   ;TABLE 1 ROTATE Field data
                                1E   02D4  9812         .Byte ^x1E
                                2C   02D5  9813         .Byte ^x2C
                                19   02D6  9814         .Byte ^x19
                                2B   02D7  9815         .Byte ^x2B
                                3B   02D8  9816         .Byte ^x3B
                                19   02D9  9817         .Byte ^x19
                                1E   02DA  9818         .Byte ^x1E
                                1A   02DB  9819         .Byte ^x1A
                                     02DC  9820
                              02D1   02DC  9821 8$:     .Word ^x2D1                  ;TABLE 2 ALPCTL Field Data
                              02C1   02DE  9822         .Word ^x2C1
                              022D   02E0  9823         .Word ^x22D
                              022A   02E2  9824         .Word ^x22A
                              026E   02E4  9825         .Word ^x26E
                              027F   02E6  9826         .Word ^x27F
                              027F   02E8  9827         .Word ^x27F
                              027C   02EA  9828         .Word ^x27C
                              026B   02EC  9829         .Word ^x26B
                                     02EE  9830
                                01   02EE  9831 9$:     .Byte   1                    ;TABLE 3 RSRC Field data
                                01   02EF  9832         .Byte   1
                                02   02F0  9833         .Byte   2
                                03   02F1  9834         .Byte   3
                                01   02F2  9835         .Byte   1
                                01   02F3  9836         .Byte   1
                                01   02F4  9837         .Byte   1
                                09   02F5  9838         .Byte   9
                                0B   02F6  9839         .Byte   ^xB
                                     02F7  9840
                                00   02F7  9841 10$:    .Byte   0                    ;TABLE 4 PSL 'C' Bit data-to be loaded
                                00   02F8  9842         .Byte   0
                                00   02F9  9843         .Byte   0
                                01   02FA  9844         .Byte   1
                                00   02FB  9845         .Byte   0
                                00   02FC  9846         .Byte   0
                                00   02FD  9847         .Byte   0
                                00   02FE  9848         .Byte   0
                                01   02FF  9849         .Byte   1
                                     0300  9850
```

```
                    01  0300  9851 11$:    .Byte   1                    ;TABLE 5 DTYPE Data to be Loaded
                    01  0301  9852          .Byte   1
                    03  0302  9853          .Byte   3
                    01  0303  9854          .Byte   1
                    00  0304  9855          .Byte   0
                    00  0305  9856          .Byte   0
                    01  0306  9857          .Byte   1
                    01  0307  9858          .Byte   1
                    01  0308  9859          .Byte   1
                        0309  9860
                80000002  0309  9861 12$:    .Long ^x80000002             ;TABLE 6 WBus data - to be checked
                FFFFFFFF  030D  9862          .Long ^xFFFFFFFF
                00000002  0311  9863          .Long   2
                C0000000  0315  9864          .Long ^xC0000000
                        0319  9865
                E0000000  0319  9866 13$:    .Long ^xE0000000             ;TABLE 7 Q Register data-to be checked
                40000000  031D  9867          .Long ^x40000000
                00000002  0321  9868          .Long   2
                80000001  0325  9869          .Long ^x80000001
                00000002  0329  9870          .Long   2
                00000102  032D  9871          .Long ^x102
                00000002  0331  9872          .Long   2
                00020006  0335  9873          .Long ^x20006
                0000C000  0339  9874          .Long ^xC000
                        033D  9875
                C0000001  033D  9876 14$:    .Long ^xC0000001         ;TABLE 8 D Register data-to be checked
                80000001  0341  9877          .Long ^x80000001
                80000001  0345  9878          .Long ^x80000001
                C0000000  0349  9879          .Long ^xC0000000
                00000002  034D  9880          .Long   2
                00000002  0351  9881          .Long   2
                00000003  0355  9882          .Long   3
                        0359  9883
                    08  0359  9884 15$:    .Byte   8              ;TABLE 9 Q Register data to be loaded - MSRC Field
                    04  035A  9885          .Byte   4
                    04  035B  9886          .Byte   4
                    04  035C  9887          .Byte   4
                    03  035D  9888          .Byte   3
                    07  035E  9889          .Byte   7
                    01  035F  9890          .Byte   1
                    06  0360  9891          .Byte   6
                    06  0361  9892          .Byte   6
                        0362  9893
                    0A  0362  9894 16$:    .Byte ^xA              ;TABLE 10 D Register data to be loaded - RSRC Field
                    08  0363  9895          .Byte   8
                    08  0364  9896          .Byte   8
                    08  0365  9897          .Byte   8
                    08  0366  9898          .Byte   8
                    04  0367  9899          .Byte   4
                    04  0368  9900          .Byte   4
                    08  0369  9901          .Byte   8
                    08  036A  9902          .Byte   8
                        036B  9903
                    0A  036B  9904 17$:    .Byte ^xA                 ;TABLE 11 RSRC Data for DIVSLOW- instruction
                    08  036C  9905          .Byte   8
```

```
                    08  036D  9906          .Byte   8
                    08  036E  9907          .Byte   8
                    08  036F  9908          .Byte   8
                    04  0370  9909          .Byte   4
                    04  0371  9910          .Byte   4
                    08  0372  9911          .Byte   8
                    08  0373  9912          .Byte   8
                        0374  9913
                        0374  9914 BEGIN_CPU_DATA
                        0002  9915
                        0002  9916 RTEMP_DATA
                        0001  9917
                00000000  0001  9918          .Long     0             ;RTEMP0 - IGNORE
                00000000  0005  9919          .Long     0             ;RTEMP1
                00000001  0009  9920          .Long     1             ;RTEMP2
                80000000  000D  9921          .Long   ^x80000000      ;RTEMP3
                C0000000  0011  9922          .Long   ^xC0000000      ;RTEMP4
                00004000  0015  9923          .Long   ^x4000          ;RTEMP5
                80004000  0019  9924          .Long   ^x80004000      ;RTEMP6
                00000040  001D  9925          .Long   ^x40            ;RTEMP7
                E0000000  0021  9926          .Long   ^xE0000000      ;RTEMP8
                7FFFFFFF  0025  9927          .Long   ^x7FFFFFFF      ;RTEMP9
                F0000000  0029  9928          .Long   ^xF0000000      ;RTEMPA
                80000001  002D  9929          .Long   ^x80000001      ;RTEMPB
                        0031  9930
                        0031  9931 END_CPU_DATA
                        0374  9932
                        0374  9933 END_TEST_DATA
                        0374
                        0374      ;-----------------------------------------------------------------------------
                        0374  9934
                        0374  9935 ENDTEST
```

```
001B        .SBTTL  "TEST   032     ALK Gate Array Test #7
001B  9937 ;*******************************************************************
001B  9938 ;++
001B  9939 ;
001B  9940 ; FUNCTIONAL DESCRIPTION
001B  9941 ;
001B  9942 ;       This test, which consists of three subtests, continues to check the ALU
001B  9943 ;       shift and Q Register shift functions of the ALK Gate Array Chip. This
001B  9944 ;       test has the LOOP Flag set.
001B  9945 ;
001B  9946 ; TEST ALGORITHM
001B  9947 ;
001B  9948 ;       SUBTEST 1
001B  9949 ;
001B  9950 ;       1. Setup loop of four iterations.
001B  9951 ;       2. Load the micro words to perform the test.
001B  9952 ;       3. Load appropriate micro fields.
001B  9953 ;       4. Execute the micro sequence.
001B  9954 ;               1. Condition the DPM with data from Tables 1 thru 11.
001B  9955 ;               2. Test the DPM with data from Tables 1 thru 11.
001B  9956 ;               3. Read WBUS into RDM.
001B  9957 ;       5. Check the results of the WBUS read.
001B  9958 ;
001B  9959 ;       SUBTEST 2
001B  9960 ;
001B  9961 ;       1. Setup loop of four iterations.
001B  9962 ;       2. Load the micro words to perform the test.
001B  9963 ;       3. Load appropriate micro fields.
001B  9964 ;       4. Execute the micro sequence.
001B  9965 ;               1. Condition the DPM with data from Tables 1 thru 11.
001B  9966 ;               2. Test the DPM with data from Tables 1 thru 11.
001B  9967 ;               3. Read WBUS into RDM.
001B  9968 ;       5. Check the results of the WBUS read (Q Register).
001B  9969 ;
001B  9970 ;       SUBTEST 3
001B  9971 ;
001B  9972 ;       1. Setup loop of four iterations.
001B  9973 ;       2. Load the micro words to perform the test.
001B  9974 ;       3. Load appropriate micro fields.
001B  9975 ;       4. Execute the micro sequence.
001B  9976 ;               1. Condition the DPM with data from Tables 1 thru 11.
001B  9977 ;               2. Test the DPM with data from Tables 1 thru 11.
001B  9978 ;               3. Read WBUS into RDM.
001B  9979 ;       5. Check the results of the WBUS read (D Register).
001B  9980 ;
001B  9981 ; DATA PATTERNS Including the initial load the D and Q Registers
001B  9982 ;
001B  9983 ;       AMUX            BMUX            QREG            DREG
001B  9984 ;
001B  9985 ;1      1               1               0               80000000
001B  9986 ;2      0               0               0               0
001B  9987 ;3      FFFFFFFE        0               40000000        0
001B  9988 ;4      FFFFFFFF        0               40000000        0
001B  9989 ;
001B  9990 ; LOGIC DESCRIPTION
```

```
001B  9991 ;
001B  9992 ;This test checks the ALK Gate Array's gates for proper operation. These
001B  9993 ;patterns were developed using the XLOG Simulator.
001B  9994 ;
001B  9995 ;The following display is the simulator output,with additions to make it more
001B  9996 ;legible. The AMUX and BMUX represent the two inputs to the ALU. The PSLC_H
001B  9997 ;input represents the PSL 'C' bit, QMSB_H represents the Most Significant Bit
001B  9998 ;of the Q Register while QR00_H represents the LSB of the Q Register. ALU31_H
001B  9999 ;represents the MSB of the ALU and ALU00_H equals the LSB of the ALU. ROT_5_H
001B 10000 ;thru ROT_0_H are the ROTATE Field of the Micro Word. OP_9_H thru OP_0_H are
001B 10001 ;the ALPCTL Field of the Micro Word. DSIZE_1_H and DSIZE_0_H are the DTYPE
001B 10002 ;Field of the Micro Word. WMUX_CONT indicates whether the WMUX Tri-State
001B 10003 ;Drivers are active for any given test. MUX, ALU and DQ CONT is the decode of
001B 10004 ;the simulator output recognizable by the Assembler. The two columns under the
001B 10005 ;ROTATE heading decode the ROTATE Field output according to the DPM Functional
001B 10006 ;Specification. The line showing ALU and some data and QREG and some data
001B 10007 ;represent the necessary data in both entities to promote a successful and
001B 10008 ;meaningful test for that particular OPCODE, DSIZE and ROTATE Field.
001B 10009 ;
001B 10010 ; LOOP FLAG IS SET AND OUTPUT IS ENABLED
001B 10011 ;
001B 10012 ;                    BOOLEAN MAP
001B 10013 ;
001B 10014 ;DD ALP QQR RRR RR 0000 0000 00 AA * AAQQQQ W
001B 10015 ;SS LOS MRO 000 00 PPPP PPPP PP LL * SSSSSS M
001B 10016 ;II UOL S0I III II |||| |||| || UU * IIIIII U
001B 10017 ;ZZ SPC B0| ||| || 9876 5432 10 30 * 000000 X
001B 10018 ;EE O|| ||5 432 10 |||| |||| || 10 * |||||| 
001B 10019 ;|| |HH HH| ||| || HHHH HHHH HH || * 303100 C
001B 10020 ;10 H     H HHH HH               HH * 101570 O
001B 10021 ;||                                * |||||| N                                    ROTATE
001B 10022 ;HH                                * LLLLLL T MUX   ALU     DQCON         ASI/QSI
001B 10023 ;*****************************************************************************
001B 10024 ;1
001B 10025 ;DSIZE_1_H 0     DSIZE_0_H 0     ALUS0_H 0
001B 10026 ;AMUX(M2)_1      BMUX(R2)_1      CI=0    LOOP_H  1
001B 10027 ;00 011 010 010 00 0001 1010 00 01 * 111111 E M.R2  AND.SR   SQL              SHF
001B 10028 ;ALU 1  QREG 1  PSL<C> 1        RESULTS: WB=0, QREG=2, DREG=0
001B 10029 ;
001B 10030 ;2
001B 10031 ;DSIZE_1_H 0     DSIZE_0_H 1     ALUS0_H 0
001B 10032 ;AMUX(M1)_0      BMUX(Q)_0       CI=PSL<C>=0     LOOP_H  1
001B 10033 ;01 010 000 011 11 0011 0110 10 00 * 111111 E M.Q2 A+B+CI.SR  SQL.D_WX      ROT
001B 10034 ;ALU 0  QREG 0  PSL<C> 0        RESULTS: WB=0, QREG=0, DREG=0
001B 10035 ;
001B 10036 ;3
001B 10037 ;DSIZE_1_H 0     DSIZE_0_H 1     ALUS0_H 0
001B 10038 ;AMUX(D)_0       BMUX(R3)_80000000       CI=1    LOOP_H  1
001B 10039 ;01 010 100 011 10 1001 0110 11 11 * 110111 E D.R2 A+B+CI.SR SQR.WX_DSR.D_WX ROT
001B 10040 ;ALU 80000001   QREG 80000000   PSL<C> 0   RESULTS: WB=40000000, QREG=40000000,
001B 10041 ;                                         RESULTS:  DREG=0
001B 10042 ;
001B 10043 ;4
001B 10044 ;DSIZE_1_H 1     DSIZE_0_H 1     ALUS0_H 0
001B 10045 ;AMUX(M1)_0      BMUX(R1)_0      CI=ALKC=0       LOOP_H 1
```

```
001B 10046 ;11 010 100 001 01 0101 1111 00 00 * 111111 E XM.R   NOTAND     NOP      1   1
001B 10047 ;ALU 0  QREG 80000000   PSL<C> 0        RESULTS: WB=0, QREG=80000000, DREG=0
001B 10048 ;
001B 10049 ; ASSUMPTIONS
001B 10050 ;
001B 10051 ;       This test requires no signals from any other CPU modules but assumes an
001B 10052 ;       operational RBUS, MBUS, WBUS, SBUS, CCC Gate array and CLA Gate array.
001B 10053 ;
001B 10054 ; ERROR DESCRIPTION
001B 10055 ;
001B 10056 ; SUBTEST 1 Data Expected    SUBTEST 2 Data Expected    SUBTEST 3 Data Expected
001B 10057 ;                WBUS                       QREG                       DREG
001B 10058 ;1               0                          2                          0
001B 10059 ;2               0                          0                          0
001B 10060 ;3               0                          40000000                   0
001B 10061 ;4               0                          80000000                   0
001B 10062 ;
001B 10063 ; Data Received: SUBTEST 1                  SUBTEST 2                  SUBTEST 3
001B 10064 ; LOOP COUNT:
001B 10065 ;--
001B 10066 ;*****************************************************************************
001B 10067
001B 10068          SUBTEST 1
001E
001E        ;/////////////////////////////////////////////////////////////////////////////
001E 10069
001E 10070          INITIALIZE                          ;DCLO low,Init scratchpads
0020 10071          LOOP    I,1,4                       ;Set loop count
0027 10072          LOADREG WDREG,8$,I                  ;Load RDM D Register for PSL 'C' bit
002F 10073          LDFIELD 12$,I,Byte,2$,64,68         ;Load MSRC Field in microword-loads Q
0040 10074          LDFIELD 30$,I,Byte,2$,34,39         ;Load RSRC Field in microword-loads D
0051 10075          LDFIELD 40$,I,Byte,20$,34,39        ;Load RSRC Field in DIVSLOW- micro-order
0062 10076          LDFIELD 7$,I,Byte,3$,34,39          ;Load RSRC Field in test microword
0073 10077          LDFIELD 14$,I,Byte,3$,40,41         ;Load DTYPE Field in test microword
0084 10078          LDFIELD 6$,I,Byte,3$,64,68          ;Load MSRC Field in test microword
0095 10079          LDFIELD 4$,I,Byte,3$,58,63          ;Load ROTATE Field in test microword
00A6 10080          LDFIELD 5$,I,Word,3$,48,57          ;Load ALPCTL Field in test microword
00B7 10081          LOADDCS 1$,,0,6                     ;Load the test micro instrs.
00BE 10082          ERRLOOP                             ;Set the error loop
00C0 10083          FETCH   0                           ;Get the first micro word
00C5 10084          BRSTCLK 5                           ;Clock stop location 5
00C9 10085          CMPREG  WHREG,9$,,Long              ;Check the WBus received data
00D2 10086          IFERROR ,<<ALK>>                    ;Continue if no error else call out ALK
00D8 10087          ENDLOOP I                           ;Continue with loop
00DB 10088
00DB 10089          SUBTEST 2
00DE
00DE        ;/////////////////////////////////////////////////////////////////////////////
00DE 10090
00DE 10091          LOOP    I,1,4                       ;Set loop count
00E5 10092          LOADREG WDREG,8$,I                  ;Load RDM D Register for PSL 'C' bit
00ED 10093          LDFIELD 12$,I,Byte,2$,64,68         ;Load MSRC Field in microword-loads Q
00FE 10094          LDFIELD 30$,I,Byte,2$,34,39         ;Load RSRC Field in microword-loads D
010F 10095          LDFIELD 40$,I,Byte,20$,34,39        ;Load RSRC Field in DIVSLOW- micro-order
0120 10096          LDFIELD 7$,I,Byte,3$,34,39          ;Load RSRC Field in test microword
```

```
                0131 10097          LDFIELD 14$,I,Byte,3$,40,41     ;Load DTYPE Field in test microword
                0142 10098          LDFIELD 6$,I,Byte,3$,64,68      ;Load MSRC Field in test microword
                0153 10099          LDFIELD 4$,I,Byte,3$,58,63      ;Load ROTATE Field in test microword
                0164 10100          LDFIELD 5$,I,Word,3$,48,57      ;Load ALPCTL Field in test microword
                0175 10101          LOADDCS 1$,,0,6                 ;Load the test micro instrs.
                017C 10102          LOADDCS 15$,,5,2                ;Overlay outputs
                0183 10103          ERRLOOP                         ;Set the error loop
                0185 10104          FETCH   0                       ;Get the first micro word
                018A 10105          BRSTCLK 6                       ;Clock stop location 6
                018E 10106          CMPREG  WHREG,10$,I,Long        ;Check the WBus received data (Q Reg)
                0197 10107          IFERROR ,<<ALK>>                ;Continue if no error else call out ALK
                019D 10108          ENDLOOP I                       ;Continue with loop
                01A0 10109
                01A0 10110          SUBTEST 3
                01A3
                01A3            ;////////////////////////////////////////////////////////////////////////////
                01A3 10111
                01A3 10112          LOOP    I,1,4                   ;Set loop count
                01AA 10113          LOADREG WDREG,8$,I              ;Load RDM D Register for PSL 'C' bit
                01B2 10114          LDFIELD 12$,I,Byte,2$,64,68     ;Load MSRC Field in microword-loads Q
                01C3 10115          LDFIELD 30$,I,Byte,2$,34,39     ;Load RSRC Field in microword-loads D
                01D4 10116          LDFIELD 40$,I,Byte,20$,34,39    ;Load RSRC Field in DIVSLOW- micro-order
                01E5 10117          LDFIELD 7$,I,Byte,3$,34,39      ;Load RSRC Field in test microword
                01F6 10118          LDFIELD 14$,I,Byte,3$,40,41     ;Load DTYPE Field in test microword
                0207 10119          LDFIELD 6$,I,Byte,3$,64,68      ;Load MSRC Field in test microword
                0218 10120          LDFIELD 4$,I,Byte,3$,58,63      ;Load ROTATE Field in test microword
                0229 10121          LDFIELD 5$,I,Word,3$,48,57      ;Load ALPCTL Field in test microword
                023A 10122          LOADDCS 1$,,0,6                 ;Load the test micro instrs.
                0241 10123          LOADDCS 16$,,5,3                ;Overlay outputs
                0248 10124          ERRLOOP                         ;Set the error loop
                024A 10125          FETCH   0                       ;Get the first micro word
                024F 10126          BRSTCLK 7                       ;Clock stop location 7
                0253 10127          CMPREG  WHREG,11$,,Long         ;Check the WBus received data (D Reg)
                025C 10128          IFERROR ,<<ALK>>                ;Continue if no error else call out ALK
                0262 10129          ENDLOOP I                       ;Continue with loop
                0265 10130          SKIP                            ;End of test
                026C 10131
                026C 10132 BEGIN_TEST_DATA
                026C
                026C            ;-----------------------------------------------------------------------------
                026C 10133 ;+
                026C 10134 ; The following micro words are used to test the ALK Gate Array.
                026C 10135 ;-
                026C 10136 ;
                026C 10137 ; Load Q register and D register with conditioning data MSRC/12$, RSRC/30$
                026C 10138 ; Then set C bit in PSL
                026C 10139 ;
                026C 10140 1$:
U 00, 7700,C800,0364,0300,0470,1801   026C 10141 ;% 00    NOP
                0277 10145 2$:
U 01, 7700,8800,0057,0304,0470,1802   0277 10146 ;% 01    ALPCTL/WX_D_R.Q_M,RSRC/TEMP1,MSRC/TEMP0
U 02, 5700,C800,0364,0300,0070,1803   0282 10150 ;% 02    WB_RDM,CCPSL/PSL_WB.CCBR_ALUS,MUX/Z.S,ALU/OR,DQ1/NOP
                028D 10154 ;
                028D 10155 ; Set LOOP flag RSRC/40$
                028D 10156 ;
```

```
                                028D 10157 20$:
U 03. 7701,8800,027E,0200,0470,1804   028D 10158 ;% 03   ALPCTL/DIVSLOW-,RSRC/TEMP0,SIZE[LONG]
                                0298 10162 ;
                                0298 10163 ; ROT/4$,ALPCTL/5$,MSRC/6$,RSRC/7$ = test micro word
                                0298 10164 ;
                                0298 10165 3$:
U 04. 6700,0800,5B64,0300,0470,1805   0298 10166 ;% 04   RDM_WB,ROT/ZERO,ALPCTL/NOP,MSRC/TEMP0,RSRC/TEMP0
U 05. 7300,4800,0364,0300,0470,1806   02A3 10170 ;% 05   NOP,STOP.CLOCK
                                02AE 10174 15$:
U 05. 6700,0800,02C7,0300,0470,1806   02AE 10175 ;% 05   RDM_WB,ALPCTL/WX_Q.Q_D          ;Read Q Register into RDM
U 06. 7300,C800,0364,0300,0470,1807   02B9 10179 ;% 06   NOP,STOP.CLOCK                 ;Stop clock and check results
                                02C4 10183 16$:
U 05. 7700,0800,02C7,0300,0470,1806   02C4 10184 ;% 05   ALPCTL/WX_Q.Q_D               ;Load D Register into Q Register
U 06. 6700,8800,02C7,0300,0470,1807   02CF 10188 ;% 06   RDM_WB,ALPCTL/WX_Q.Q_D        ;Read Q(D) Register into RDM
U 07. 7300,C800,0364,0300,0470,1808   02DA 10192 ;% 07   NOP,STOP.CLOCK                 ;Stop clock and check results
                                02E5 10196
                             08 02E5 10197 4$:     .Byte ^x08                       ;TABLE 1 ROTATE Field data
                             2C 02E6 10198         .Byte ^x2C
                             0F 02E7 10199         .Byte ^x0F
                             0E 02E8 10200         .Byte ^x0E
                                02E9 10201
                           0068 02E9 10202 5$:     .Word ^x068                      ;TABLE 2 ALPCTL Field Data
                           00DA 02EB 10203         .Word ^x0DA
                           025B 02ED 10204         .Word ^x25B
                           017C 02EF 10205         .Word ^x17C
                                02F1 10206
                             02 02F1 10207 6$:     .Byte   2                        ;TABLE 3 MSRC Field data
                             01 02F2 10208         .Byte   1
                             01 02F3 10209         .Byte   1
                             01 02F4 10210         .Byte   1
                                02F5 10211
                             02 02F5 10212 7$:     .Byte   2                        ;TABLE 4 RSRC Field data
                             01 02F6 10213         .Byte   1
                             01 02F7 10214         .Byte   1
                             01 02F8 10215         .Byte   1
                                02F9 10216
                             01 02F9 10217 8$:     .Byte   1                        ;TABLE 5 PSL 'C' Bit data-to be loaded
                             00 02FA 10218         .Byte   0
                             00 02FB 10219         .Byte   0
                             00 02FC 10220         .Byte   0
                                02FD 10221
                       00000000 02FD 10222 9$:     .Long   0                        ;TABLE 6 WBus data - to be checked
                                0301 10223
                       00000002 0301 10224 10$:    .Long   2                        ;TABLE 7 Q Register data-to be checked
                       00000000 0305 10225         .Long   0
                       40000000 0309 10226         .Long ^X40000000
                       80000000 030D 10227         .Long ^x80000000
                                0311 10228
                       00000000 0311 10229 11$:    .Long   0       ;TABLE 8 D Register data - to be checked
                                0315 10230
                             01 0315 10231 12$:    .Byte   1       ;TABLE 9 Q Register data to be loaded - MSRC Field
                             01 0316 10232         .Byte   1
                             05 0317 10233         .Byte   5
                             05 0318 10234         .Byte   5
                                0319 10235
```

```
                     03  0319 10236 30$:     .Byte   3       ;TABLE 12 D Register data to be loaded - RSRC Field
                     01  031A 10237          .Byte   1
                     01  031B 10238          .Byte   1
                     01  031C 10239          .Byte   1
                         031D 10240
                     00  031D 10241 14$:     .Byte   0       ;TABLE 11 DTYPE Data to be Loaded
                     01  031E 10242          .Byte   1
                     01  031F 10243          .Byte   1
                     03  0320 10244          .Byte   3
                         0321 10245
                     03  0321 10246 40$:     .Byte   3       ;TABLE 13 RSRC Data for DIVSLOW- Micro-order
                     01  0322 10247          .Byte   1
                     01  0323 10248          .Byte   1
                     01  0324 10249          .Byte   1
                         0325 10250
                         0325 10251 BEGIN_CPU_DATA
                         0002 10252
                         0002 10253 RTEMP_DATA
                         0001 10254
                00000000 0001 10255          .Long     0             ;RTEMP0 - IGNORE
                00000000 0005 10256          .Long     0             ;RTEMP1
                00000001 0009 10257          .Long     1             ;RTEMP2
                80000000 000D 10258          .Long   ^x80000000      ;RTEMP3
                C0000000 0011 10259          .Long   ^xC0000000      ;RTEMP4
                40000000 0015 10260          .Long   ^x40000000      ;RTEMP5
                         0019 10261
                         0019 10262 END_CPU_DATA
                         0325 10263
                         0325 10264 END_TEST_DATA
                         0325
                         0325       ;-----------------------------------------------------------------------
                         0325 10265
                         0325 10266 ENDTEST
```

```
002D                    .SBTTL  "TEST   033     Rotator Test  1 SBUS Data Integrity Test
002D 10268 ;*****************************************************************************
002D 10269 ;++
002D 10270 ;
002D 10271 ; FUNCTIONAL DESCRIPTION
002D 10272 ;
002D 10273 ;       This test passes 12 patterns through the Rotator to examine the SBUS.
002D 10274 ;
002D 10275 ; TEST ALGORITHM
002D 10276 ;
002D 10277 ;       This test is implemented in two subtests. The first subtest passes 6
002D 10278 ;       stuck at zero or one test patterns unrotated through the SBUS, while
002D 10279 ;       the second subtest passes the same patterns rotated right by 0, 1, 2
002D 10280 ;       and 3 bits through the SBUS to check the connections to the ALP slices.
002D 10281 ;       The algorithms for each subtest are specified below:
002D 10282 ;
002D 10283 ;       Subtest 1
002D 10284 ;
002D 10285 ;       1. Setup a loop of six iterations.
002D 10286 ;       2. Generate a stuck at 1 or 0 pattern.
002D 10287 ;       3. Load the WDREG with the pattern.
002D 10288 ;       4. Execute the micro-sequence:
002D 10289 ;               1. Zero out the PLATCH.
002D 10290 ;               2. Load RTEMP0 with the test pattern.
002D 10291 ;               3. Source RTEMP0 onto the SBUS, pass it directly from the
002D 10292 ;                  ALU BMUX to the WBUS and read it back into the RDM.
002D 10293 ;       5. Check the results of the test.
002D 10294 ;       6. Loop back until done.
002D 10295 ;
002D 10296 ;       NOTE -  Micro-sequence step 3 uses the ALPCTL/WX_S.Q_0 micro-order to
002D 10297 ;               bypass the ALP ALU.
002D 10298 ;
002D 10299 ;       Subtest 2
002D 10300 ;
002D 10301 ;        1. Start signature analysis.
002D 10302 ;        2. Load the K index with 1.
002D 10303 ;        3. Set up a loop of 4 iterations on I.
002D 10304 ;        4. Prepare the micro-instruction at 10$ to load the PLATCH with a test
002D 10305 ;           value from the table at 3$ indexed by I.
002D 10306 ;        5. Load the test micro-instruction into DCS.
002D 10307 ;        6. Set up a loop of 6 iterations on J.
002D 10308 ;        7. Based on the value of J, generate a standard stuck at zero or one
002D 10309 ;           test pattern and place it into 1$.
002D 10310 ;        8. Load the test pattern from 1$ into WDREG.
002D 10311 ;        9. Loop on error from here.
002D 10312 ;       10. Execute the micro-code to:
002D 10313 ;               1. Load the PLATCH with the value given in step 4.
002D 10314 ;               2. Load RTEMP0 with the data in the WDREG.
002D 10315 ;               3. Execute a ROT/RR.RR.P micro-order and read the result back
002D 10316 ;                  to the WHREG.
002D 10317 ;       11. Compare the data in the WHREG with the expected data from the table
002D 10318 ;           at 2$ indexed by K.
002D 10319 ;       12. If the expected and received data is not the same, issue an error.
002D 10320 ;       13. Increment the index K.
002D 10321 ;       14. Loop back to step 6 until all six patterns have been tried.
```

```
002D 10322 ;       15. Loop back to step 3 until all four PLATCH values have been tried.
002D 10323 ;       16. Stop signature analysis.
002D 10324 ;
002D 10325 ; TEST DATA
002D 10326 ;
002D 10327 ;       For Subtest 1 six stuck at zero or one test patterns are used:
002D 10328 ;
002D 10329 ;              AAAAAAAA
002D 10330 ;              55555555
002D 10331 ;              33333333
002D 10332 ;              0F0F0F0F
002D 10333 ;              00FF00FF
002D 10334 ;              0000FFFF
002D 10335 ;
002D 10336 ;       For Subtest 2 the same six stuck at zero or one test patterns are right
002D 10337 ;       rotated by 0, 1, 2 and 3 bits:
002D 10338 ;
002D 10339 ;       I J K   PLATCH  RBUS DATA       WBUS DATA
002D 10340 ;
002D 10341 ;       1 1 1   0       AAAAAAAA        AAAAAAAA
002D 10342 ;       1 2 2   0       55555555        55555555
002D 10343 ;       1 3 3   0       33333333        33333333
002D 10344 ;       1 4 4   0       0F0F0F0F        0F0F0F0F
002D 10345 ;       1 5 5   0       00FF00FF        00FF00FF
002D 10346 ;       1 6 6   0       0000FFFF        0000FFFF
002D 10347 ;       2 1 7   1       AAAAAAAA        55555555
002D 10348 ;       2 2 8   1       55555555        AAAAAAAA
002D 10349 ;       2 3 9   1       33333333        99999999
002D 10350 ;       2 4 A   1       0F0F0F0F        87878787
002D 10351 ;       2 5 B   1       00FF00FF        807F807F
002D 10352 ;       2 6 C   1       0000FFFF        80007FFF
002D 10353 ;       3 1 D   2       AAAAAAAA        AAAAAAAA
002D 10354 ;       3 2 E   2       55555555        55555555
002D 10355 ;       3 3 F   2       33333333        CCCCCCCC
002D 10356 ;       3 4 10  2       0F0F0F0F        C3C3C3C3
002D 10357 ;       3 5 11  2       00FF00FF        C03FC03F
002D 10358 ;       3 6 12  2       0000FFFF        C0003FFF
002D 10359 ;       4 1 13  3       AAAAAAAA        55555555
002D 10360 ;       4 2 14  3       55555555        AAAAAAAA
002D 10361 ;       4 3 15  3       33333333        66666666
002D 10362 ;       4 4 16  3       0F0F0F0F        E1E1E1E1
002D 10363 ;       4 5 17  3       00FF00FF        E01FE01F
002D 10364 ;       4 6 18  3       0000FFFF        E0001FFF
002D 10365 ;
002D 10366 ; LOGIC DESCRIPTION
002D 10367 ;
002D 10368 ;       The Rotator's SBUS is checked for data integrity. The SBUS is generated
002D 10369 ;       by the SRM gate array bit slices and is sent to the ALP bit slices,
002D 10370 ;       where it can be driven out onto the WBUS.
002D 10371 ;
002D 10372 ; ASSUMPTIONS
002D 10373 ;
002D 10374 ;       This test requires an operational RBUS, SBUS and WBUS.
002D 10375 ;       The ALP special function WMUX <- BMUX must also work.
002D 10376 ;
```

```
                     002D 10377 ; ERROR DESCRIPTION
                     002D 10378 ;
                     002D 10379 ;       Subtest 1 will call out the SRM, SRK and ALP gate arrays and the data:
                     002D 10380 ;
                     002D 10381 ;       EXPECTED SBUS DATA
                     002D 10382 ;       RECEIVED SBUS DATA
                     002D 10383 ;       LOOP COUNT
                     002D 10384 ;
                     002D 10385 ;       Subtest 2 will call out the ALP and SRK gate arrays and the data:
                     002D 10386 ;
                     002D 10387 ;       EXPECTED SBUS DATA
                     002D 10388 ;       RECEIVED SBUS DATA
                     002D 10389 ;       PLATCH VALUE USED (INDEX I)
                     002D 10390 ;       TEST PATTERN USED (INDEX J)
                     002D 10391 ;       INDEX TO EXPECTED DATA TABLE (INDEX K)
                     002D 10392 ;       RBUS INPUT DATA (RTEMP0)
                     002D 10393 ;       PLATCH DATA (MTEMP1)
                     002D 10394 ;
                     002D 10395 ;--
                     002D 10396 ;*******************************************************************************
                     002D 10397
                     002D 10398         SUBTEST         1
                     0030
                     0030       ;////////////////////////////////////////////////////////////////////////////
                     0030 10399 ;+
                     0030 10400 ; This subtest examines the rotator's SBUS without shifting.
                     0030 10401 ;-
                     0030 10402         INITIALIZE                              ; INITIALIZE THE CPU
                     0032 10403         LOADDCS         10$,,0,7                ; LOAD TEST INSTRS. INTO DCS
                     0039 10404         LOOP            I,1,6                   ; LOOP FOR 6 SA01 PATTERNS
                     0040 10405         SA01PAT         I,1$                    ; GENERATE A SA01 TEST PATTERN
                     0048 10406         LOADREG         WDREG,1$,,L             ; LOAD THE PATTERN INTO WDREG
                     0050 10407         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                     0052 10408         FETCH           0                       ; SEND THE PATTERN THROUGH THE
                     0057 10409         BRSTCLK         6                       ; ROTATOR TO THE RDM
                     005B 10410         CMPREG          WHREG,1$,,L             ; CHECK THE RESULT
                     0064 10411         IFERROR         ,<<SRM><SRK><ALP>>      ; ERROR CALLOUT
                     006C 10412         ENDLOOP         I                       ; CONTINUE LOOP
                     006F 10413
                     006F 10414         SUBTEST         2
                     0072
                     0072       ;////////////////////////////////////////////////////////////////////////////
                     0072 10415 ;+
                     0072 10416 ; This subtest examines the SBUS inputs to second level shifter on ALP.
                     0072 10417 ;-
                     0072 10418         LDINDEX         K,1                     ; INITIALIZE THE INDEX K
                     0079 10419         LOADDCS         11$,,0,6                ; LOAD TEST INSTRS. INTO DCS
                     0080 10420         LOOP            I,1,4                   ; LOOP ON I FOR 4 PLATCH VALUES
                     0087 10421         LDFIELD         3$,I,W,12$,31,39        ; LITRL FIELD GETS PLATCH VALUE
                     0098 10422         LOADDCS         12$,,1,1                ; LOAD PL_[LITRL] INTO DCS
                     009F 10423         LOOP            J,1,6                   ; LOOP ON J FOR 6 TEST PATTERNS
                     00A6 10424         SA01PAT         J,1$                    ; GENERATE A SA01 TEST PATTERN
                     00AE 10425         LOADREG         WDREG,1$,,L             ; LOAD TEST PATTERN INTO WDREG
                     00B6 10426         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                     00B8 10427         FETCH           0                       ; EXECUTE THE ROT/RR.RR.P ORDER
```

```
                                  00BD          BRSTCLK        4                          ; AND WRITE RESULT TO WHREG
                                  00C1          CMPREG         WHREG,2$,K,L               ; IS WHREG CORRECT?
                                  00CA          IFERROR        2,<<ALP><SRK>>             ; IF NOT, SIGNAL AN ERROR
                                  00D1          INCINDEX       K                          ; INCREMENT INDEX K
                                  00D4          ENDLOOP        J                          ; CONTINUE SA01 PATTERNS
                                  00D7          ENDLOOP        I                          ; CONTINUE PLATCH VALUES
                                  00DA          SKIP
                                  00E1
                                  00E1  BEGIN_TEST_DATA
                                  00E1
                                  00E1      ;-----------------------------------------------------------------------
                                  00E1
                        00000000  00E1  1$:     .LONG   0       ; LOCATION TO HOLD SA01 PATTERN
                                  00E5
                                  00E5  ; Expected data for Subtest 2
                                  00E5  ; The numbers on the right indicate values for inoex K.
                                  00E5
        33333333 55555555 AAAAAAAA  00E5  2$:     .LONG   ^XAAAAAAAA,^X55555555,^X33333333        ; 01 - 03
        0000FFFF 00FF00FF 0F0F0F0F  00F1          .LONG   ^X0F0F0F0F,^X00FF00FF,^X0000FFFF        ; 04 - 06
        99999999 AAAAAAAA 55555555  00FD          .LONG   ^X55555555,^XAAAAAAAA,^X99999999        ; 07 - 09
        80007FFF 807F807F 87878787  0109          .LONG   ^X87878787,^X807F807F,^X80007FFF        ; 0A - 0C
        CCCCCCCC 55555555 AAAAAAAA  0115          .LONG   ^XAAAAAAAA,^X55555555,^XCCCCCCCC        ; 0D - 0F
        C0003FFF C03FC03F C3C3C3C3  0121          .LONG   ^XC3C3C3C3,^XC03FC03F,^XC0003FFF        ; 10 - 12
        66666666 AAAAAAAA 55555555  012D          .LONG   ^X55555555,^XAAAAAAAA,^X66666666        ; 13 - 15
        E0001FFF E01FE01F E1E1E1E1  0139          .LONG   ^XE1E1E1E1,^XE01FE01F,^XE0001FFF        ; 16 - 18
                                  0145
                                  0145  ; PLATCH values for Subtest 2
                                  0145
                            0000  0145  3$:     .WORD   0       ; I = 1
                            0001  0147          .WORD   1       ; I = 2
                            0002  0149          .WORD   2       ; I = 3
                            0003  014B          .WORD   3       ; I = 4
                                  014D
                                  014D  10$:
U 00, 7700,C800,0364,0300,0470,1801  014D  ;% 00   NOP                                        ; GET CLOCK UP TO SPEED
U 01, 7700,3800,7FFF,FFEF,8470,1802  0158  ;% 01   LONLIT_[20]                                ; HEX 20 = 32 DECIMAL
U 02, 7700,4800,B647,03D4,0470,1803  0163  ;% 02   RSRC/LONLIT,ALPCTL/WX_R.Q_D,ROT/PL.SL_WB   ; SLATCH = 32 BIT FIELD
U 03, 7700,8800,BE47,03D8,0470,1804  016E  ;% 03   RSRC/ZERO,ALPCTL/WX_R.Q_D,ROT/SL.PL_WB     ; PLATCH = 0
U 04, 5700,0840,0364,0300,0470,1805  0179  ;% 04   R[TEMP0]_RDM                               ; RTEMP0 = TEST PATTERN
U 05, 6700,4800,0B47,0300,0470,1806  0184  ;% 05   RSRC/TEMP0,ROT/XZ.RR,ALPCTL/WX_S.Q_0,RDM_WB ; TEST ROTATOR
U 06, 7300,C800,0364,0300,0470,1807  018F  ;% 06   STOP.CLOCK                                 ; HALT
                                  019A
                                  019A  11$:
U 00, 7700,C800,0364,0300,0470,1801  019A  ;% 00   NOP                                        ; GET CLOCK UP TO SPEED
                                  01A5  12$:
U 01, 7700,1800,EF64,0300,0470,1802  01A5  ;% 01   PL_[0]                                     ; PLATCH GETS LITRL
U 02, 5700,0840,0364,0300,0470,1803  01B0  ;% 02   R[TEMP0]_RDM                               ; RTEMP0 = TEST PATTERN
U 03, 6700,8800,1B47,0300,0470,1804  01BB  ;% 03   RSRC/TEMP0,ROT/RR.RR.P,ALPCTL/WX_S.Q_0,RDM_WB ; TEST 2ND LEVEL
U 04, 7700,4861,B370,0300,0470,1805  01C6  ;% 04   M[TEMP1]_PL                                ; READ BACK PLATCH
U 05, 7300,4800,0364,0300,0470,1806  01D1  ;% 05   STOP.CLOCK                                 ; HALT
                                  01DC
                                  01DC  END_TEST_DATA
                                  01DC
                                  01DC      ;-----------------------------------------------------------------------
                                  01DC
```

```
                    01DC 10518 ENDTEST
```

```
0020        .SBTTL  "TEST   034     Rotator Test  2 PLATCH Test
0020 10520 ;*******************************************************************************
0020 10521 ;++
0020 10522 ;
0020 10523 ; FUNCTIONAL DESCRIPTION
0020 10524 ;
0020 10525 ;       This test checks the rotator's position latch (PLATCH).
0020 10526 ;
0020 10527 ; TEST ALGORITHM
0020 10528 ;
0020 10529 ;       1. Set up a loop of six iterations.
0020 10530 ;       2. Generate a stuck at zero or one pattern.
0020 10531 ;       3. Load the pattern into the RDM WDREG.
0020 10532 ;       4. Execute the micro-sequence:
0020 10533 ;               1. Write the PLATCH from the RDM's WDREG.
0020 10534 ;               2. Read the PLATCH back into the RDM's WHREG.
0020 10535 ;       5. Check the results of the PLATCH test.
0020 10536 ;       6. Continue the loop until all the patterns have been tried.
0020 10537 ;
0020 10538 ; TEST DATA
0020 10539 ;
0020 10540 ;       This test uses the lower six bits of the standard stuck at zero or one
0020 10541 ;       test patterns.
0020 10542 ;
0020 10543 ;           Pattern Generated           Expected PLATCH contents
0020 10544 ;
0020 10545 ;               AAAAAAAA                        2A
0020 10546 ;               55555555                        15
0020 10547 ;               33333333                        33
0020 10548 ;               0F0F0F0F                        0F
0020 10549 ;               00FF00FF                        3F
0020 10550 ;               0000FFFF                        3F
0020 10551 ;
0020 10552 ; LOGIC DESCRIPTION
0020 10553 ;
0020 10554 ;       The PLATCH and its read/write control logic is inside the SRK chip.
0020 10555 ;
0020 10556 ; ASSUMPTIONS
0020 10557 ;
0020 10558 ;       This test requires an operational SBUS and WBUS bits <5:0>.
0020 10559 ;       The ALPCTL special function WMUX <- BMUX must also work.
0020 10560 ;
0020 10561 ; ERROR DESCRIPTION
0020 10562 ;
0020 10563 ;       On error this test will call out the SRK gate array.
0020 10564 ;       It will also type out the following information:
0020 10565 ;
0020 10566 ;       EXPECTED PLATCH DATA
0020 10567 ;       RECEIVED PLATCH DATA
0020 10568 ;       LOOP COUNT
0020 10569 ;
0020 10570 ;--
0020 10571 ;*******************************************************************************
0020 10572 ;///////////////////////////////////////////////////////////////////////////////
0020 10573
```

```
                        0020 10574          INITIALIZE                          ; INITIALIZE THE CPU
                        0022 10575          LOOP            I,1,6               ; LOOP FOR 6 SA01 PATTERNS
                        0029 10576          SA01PAT         I,1$                ; GENERATE A SA01 TEST PATTERN
                        0031 10577          LOADREG         WDREG,1$,,L         ; LOAD THE PATTERN INTO WDREG
                        0039 10578          ERRLOOP                             ; LOOP ON ERROR FROM HERE
                        003B 10579          FETCH           0                   ; SEND THE PATTERN THROUGH THE
                        0040 10580          BRSTCLK         3                   ; ROTATOR'S PLATCH TO THE RDM
                        0044 10581          CMPREGMSK       WHREG,2$,I,3$,,B    ; CHECK THE RESULT
                        0050 10582          IFERROR         ,<SRK>              ; ERROR CALLOUT
                        0056 10583          ENDLOOP         I                   ; CONTINUE LOOP
                        0059 10584          SKIP                                ; END OF TEST
                        0060 10585
                        0060 10586 BEGIN_TEST_DATA
                        0060
                        0060        ;-------------------------------------------------------------------------
                        0060 10587
                00000000 0060 10588 1$:     .LONG   0       ; LOCATION TO HOLD SA01 PATTERN
                        0064 10589
                      2A 0064 10590 2$:     .BYTE   ^X2A    ; EXPECTED PLATCH DATA
                      15 0065 10591         .BYTE   ^X15
                      33 0066 10592         .BYTE   ^X33
                      0F 0067 10593         .BYTE   ^X0F
                      3F 0068 10594         .BYTE   ^X3F
                      3F 0069 10595         .BYTE   ^X3F
                        006A 10596
                      3F 006A 10597 3$:     .BYTE   ^X3F    ; MASK FOR DATA COMPARISON
                        006B 10598
                        006B 10599 BEGIN_CPU_DATA
                        0002 10600
                        0002 10601 DCS_DATA
                        0001 10602
U 00, 7700,C800,0364,0300,0470,1801 0001 10603 ;% 00   NOP                         ; GET CPU UP TO SPEED
U 01, 5700,8800,BF64,0300,0470,1802 000C 10607 ;% 01   WB_RDM,ROT/SL.PL_WB        ; PLATCH = TEST PATTERN
U 02, 6700,4800,B347,0300,0470,1803 0017 10611 ;% 02   ROT/PL,ALPCTL/WX_S.Q_0,RDM_WB ; READ BACK THE PLATCH
U 03, 7300,C800,0364,0300,0470,1804 0022 10615 ;% 03   STOP.CLOCK                 ; HALT
                        002D 10619
                        002D 10620 END_CPU_DATA
                        006B 10621
                        006B 10622 END_TEST_DATA
                        006B
                        006B        ;-------------------------------------------------------------------------
                        006B 10623
                        006B 10624 ENDTEST
```

```
0020        .SBTTL  "TEST   035     Rotator Test  3 SLATCH Test
0020 10626 ;*******************************************************************************
0020 10627 ;++
0020 10628 ;
0020 10629 ; FUNCTIONAL DESCRIPTION
0020 10630 ;
0020 10631 ;       This test examines the rotator's size latch (SLATCH).
0020 10632 ;
0020 10633 ; TEST ALGORITHM
0020 10634 ;
0020 10635 ;       1. Set up a loop of six iterations.
0020 10636 ;       2. Generate a stuck at zero or one pattern.
0020 10637 ;       3. Load the pattern into the RDM WDREG.
0020 10638 ;       4. Execute the micro-sequence:
0020 10639 ;               1. Write the SLATCH from the RDM's WDREG.
0020 10640 ;               2. Read the SLATCH back into the RDM's WHREG.
0020 10641 ;       5. Check the results of the SLATCH test.
0020 10642 ;       6. Continue the loop until all the patterns have been tried.
0020 10643 ;
0020 10644 ; TEST DATA
0020 10645 ;
0020 10646 ;       This test uses the lower six bits of the standard stuck at zero or one
0020 10647 ;       test patterns.
0020 10648 ;
0020 10649 ;           Pattern Generated           Expected SLATCH contents
0020 10650 ;
0020 10651 ;               AAAAAAAA                        2A
0020 10652 ;               55555555                        15
0020 10653 ;               33333333                        33
0020 10654 ;               0F0F0F0F                        0F
0020 10655 ;               00FF00FF                        3F
0020 10656 ;               0000FFFF                        3F
0020 10657 ;
0020 10658 ; LOGIC DESCRIPTION
0020 10659 ;
0020 10660 ;       The SLATCH and its read/write control logic is inside the SRK chip.
0020 10661 ;
0020 10662 ; ASSUMPTIONS
0020 10663 ;
0020 10664 ;       This test requires an operational SBUS and WBUS bits <5:0>.
0020 10665 ;       The ALPCTL special function WMUX <- BMUX must also work.
0020 10666 ;
0020 10667 ; ERROR DESCRIPTION
0020 10668 ;
0020 10669 ;       On error this test will call out the SRK gate array.
0020 10670 ;       It will also type out the following information:
0020 10671 ;
0020 10672 ;       EXPECTED SLATCH DATA
0020 10673 ;       RECEIVED SLATCH DATA
0020 10674 ;       LOOP COUNT
0020 10675 ;
0020 10676 ;--
0020 10677 ;*******************************************************************************
0020 10678 ;///////////////////////////////////////////////////////////////////////////////
0020 10679
```

```
                        0020 10680          INITIALIZE                                  ; INITIALIZE THE CPU
                        0022 10681          LOOP            I,1,6                       ; LOOP FOR 6 SA01 PATTERNS
                        0029 10682          SA01PAT         I,1$                        ; GENERATE A SA01 TEST PATTERN
                        0031 10683          LOADREG         WDREG,1$,,L                 ; LOAD THE PATTERN INTO WDREG
                        0039 10684          ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                        003B 10685          FETCH           0                           ; SEND THE PATTERN THROUGH THE
                        0040 10686          BRSTCLK         3                           ; ROTATOR'S SLATCH TO THE RDM
                        0044 10687          CMPREGMSK       WHREG,2$,I,3$,,B            ; CHECK THE RESULT
                        0050 10688          IFERROR         ,<SRK>                      ; ERROR CALLOUT
                        0056 10689          ENDLOOP         I                           ; CONTINUE LOOP
                        0059 10690          SKIP                                        ; END OF TEST
                        0060 10691
                        0060 10692 BEGIN_TEST_DATA
                        0060
                        0060        ;-------------------------------------------------------------------------
                        0060 10693
               00000000 0060 10694 1$:      .LONG   0           ; LOCATION TO HOLD SA01 PATTERN
                        0064 10695
                     2A 0064 10696 2$:      .BYTE   ^X2A        ; EXPECTED SLATCH DATA
                     15 0065 10697          .BYTE   ^X15
                     33 0066 10698          .BYTE   ^X33
                     0F 0067 10699          .BYTE   ^X0F
                     3F 0068 10700          .BYTE   ^X3F
                     3F 0069 10701          .BYTE   ^X3F
                        006A 10702
                     3F 006A 10703 3$:      .BYTE   ^X3F        ; MASK FOR DATA COMPARISON
                        006B 10704
                        006B 10705 BEGIN_CPU_DATA
                        0002 10706
                        0002 10707 DCS_DATA
                        0001 10708
U 00, 7700,C800,0364,0300,0470,1801     0001 10709 ;% 00    NOP                         ; GET CPU UP TO SPEED
U 01, 5700,C800,B764,0300,0470,1802     000C 10713 ;% 01    WB_RDM,ROT/PL.SL_WB         ; SLATCH = TEST PATTERN
U 02, 6700,0800,BB47,0300,0470,1803     0017 10717 ;% 02    ROT/SL,ALPCTL/WX_S.Q_0,RDM_WB ; READ SLATCH BACK TO RDM
U 03, 7300,C800,0364,0300,0470,1804     0022 10721 ;% 03    STOP.CLOCK                  ; HALT
                        002D 10725
                        002D 10726 END_CPU_DATA
                        006B 10727
                        006B 10728 END_TEST_DATA
                        006B
                        006B        ;-------------------------------------------------------------------------
                        006B 10729
                        006B 10730 ENDTEST
```

```
0032        .SBTTL  "TEST   036     Rotator Test  4 PLATCH 4:3 gets WBUS 1:0 Test
0032 10732 ;*******************************************************************************
0032 10733 ;++
0032 10734 ;
0032 10735 ; FUNCTIONAL DESCRIPTION
0032 10736 ;
0032 10737 ;       This test checks that PLATCH<4:3> is loaded with WBUS<1:0> while
0032 10738 ;       PLATCH<5, 2:0> is unchanged.
0032 10739 ;
0032 10740 ; TEST ALGORITHM
0032 10741 ;
0032 10742 ;       1. Setup a loop of six iterations.
0032 10743 ;       2. Generate a stuck at 1 or 0 pattern.
0032 10744 ;       3. Load the WDREG with the pattern.
0032 10745 ;       4. Execute the micro-sequence:
0032 10746 ;               1. Zero out the PLATCH.
0032 10747 ;               2. Load PLATCH<4:3> with the test pattern.
0032 10748 ;               3. Read the PLATCH from the rotator to the RDM.
0032 10749 ;       5. Check the results of the test.
0032 10750 ;       6. Loop back until done.
0032 10751 ;
0032 10752 ; TEST DATA
0032 10753 ;
0032 10754 ;       This test uses the lower six bits of the standard stuck at zero or one
0032 10755 ;       test patterns.
0032 10756 ;
0032 10757 ;           Pattern Generated           Expected PLATCH contents
0032 10758 ;
0032 10759 ;               AAAAAAAA                        10
0032 10760 ;               55555555                        08
0032 10761 ;               33333333                        18
0032 10762 ;               0F0F0F0F                        18
0032 10763 ;               00FF00FF                        18
0032 10764 ;               0000FFFF                        18
0032 10765 ;
0032 10766 ; LOGICAL DESCRIPTION
0032 10767 ;
0032 10768 ;       This test uses logic inside the SRK gate array.
0032 10769 ;
0032 10770 ; ASSUMPTIONS
0032 10771 ;
0032 10772 ;       This test requires an operational SBUS and WBUS bits <5:0>.
0032 10773 ;       The ALPCTL special function WMUX <- BMUX must also work.
0032 10774 ;
0032 10775 ; ERROR DESCRIPTION
0032 10776 ;
0032 10777 ;       On error this test will call out the SRK gate array.
0032 10778 ;       It will also type out the following information:
0032 10779 ;
0032 10780 ;       EXPECTED PLATCH CONTENTS
0032 10781 ;       RECEIVED PLATCH CONTENTS
0032 10782 ;       LOOP COUNT
0032 10783 ;
0032 10784 ;--
0032 10785 ;*******************************************************************************
```

```
                        0032 10786 ;////////////////////////////////////////////////////////////////////////////////
                        0032 10787
                        0032 10788          INITIALIZE                                  ; INITIALIZE THE CPU
                        0034 10789          LOOP            I,1,6                       ; LOOP FOR 6 SA01 PATTERNS
                        003B 10790          SA01PAT         I,1$                        ; GENERATE A SA01 TEST PATTERN
                        0043 10791          LOADREG         WDREG,1$,,L                 ; LOAD THE PATTERN INTO WDREG
                        004B 10792          ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                        004D 10793          FETCH           0                           ; SEND THE PATTERN THROUGH THE
                        0052 10794          BRSTCLK         4                           ; ROTATOR TO THE RDM
                        0056 10795          CMPREGMSK       WHREG,2$,I,3$,,B            ; CHECK THE RESULT
                        0062 10796          IFERROR         ,<SRK>                      ; ERROR CALLOUT
                        0068 10797          ENDLOOP         I                           ; CONTINUE LOOP
                        006B 10798          SKIP                                        ; END OF TEST
                        0072 10799
                        0072 10800 BEGIN_TEST_DATA
                        0072
                        0072        ;-------------------------------------------------------------------------
                        0072 10801
               00000000 0072 10802 1$:      .LONG   0       ; LOCATION TO HOLD SA01 PATTERN
                        0076 10803
                     10 0076 10804 2$:      .BYTE   ^X10    ; EXPECTED SLATCH DATA
                     08 0077 10805          .BYTE   ^X08
                     18 0078 10806          .BYTE   ^X18
                     18 0079 10807          .BYTE   ^X18
                     18 007A 10808          .BYTE   ^X18
                     18 007B 10809          .BYTE   ^X18
                        007C 10810
                     3F 007C 10811 3$:      .BYTE   ^X3F    ; MASK FOR DATA COMPARISON
                        007D 10812
                        007D 10813 BEGIN_CPU_DATA
                        0002 10814
                        0002 10815 DCS_DATA
                        0001 10816
U 00, 7700,C800,0364,0300,0470,1801 0001 10817 ;% 00   NOP                             ; GET CPU UP TO SPEED
U 01, 7700,1800,EF64,0300,0470,1802 000C 10821 ;% 01   PL_[0]                          ; ZERO THE PLATCH
U 02, 5700,4800,FF64,0300,0470,1803 0017 10825 ;% 02   WB_RDM,ROT/OLIT0.PL43_WB        ; LOAD PLATCH WITH TEST PATTERN
U 03, 6700,C800,B347,0300,0470,1804 0022 10829 ;% 03   ROT/PL,ALPCTL/WX_S.Q_0,RDM_WB   ; READ PLATCH BACK TO RDM
U 04, 7300,4800,0364,0300,0470,1805 002D 10833 ;% 04   STOP.CLOCK                       ; HALT
                        0038 10837
                        0038 10838 END_CPU_DATA
                        007D 10839
                        007D 10840 END_TEST_DATA
                        007D
                        007D        ;-------------------------------------------------------------------------
                        007D 10841
                        007D 10842 ENDTEST
```

```
                0000 10844 .END
```

```
$CONTROL_C_CODE= 0000000A
$FILE_NOT_FOUND= 00000009
$STN            = 00000000
$TEST_NUMBER    = 00000037
ACV             = 0000000F
AC_LOW          = 00000080
ADD             = 00000019
ADDR_MATCH_HI   = 00007405
ADDR_MATCH_LO   = 00007404
ADK             = 0000000B
ALK             = 00000000
ALP             = 00000016
ARG_CNT         = 00000063 R      A9
A_REG_BYTE_0    = 00007410
A_REG_BYTE_1    = 00007411
A_REG_BYTE_2    = 00007412
A_REG_BYTE_3    = 00007413
B               = 00000001
BADD1           = 00007426
BELL_FLAG       = 00000008
BELL_KEY        = 0000000E
BUFF_ADDR_LOW   = 00007426
BURST_STOP_FLAG= 00000080
BYTE            = 00000001
CACHE_FLAG      = 00000000
CACHE_FLAG1     = 00000000
CACHE_LENGTH    = 00000001
CAK             = 0000000A
CCC             = 00000004
CCS             = 00000005
CF_KEY          = 00000024
CLA             = 00000003
CLEAR_CTRL_FILE= 00000002
CLKDCS          = 00007427
CLK_CTRL_0      = 00000020
CLK_CTRL_0_RO   = 00000040
CLK_CTRL_1      = 00000040
CLK_CTRL_1_RO   = 00000080
CLOCK_STOPED    = 00000080
CMI             = 00000007
CMICTL          = 0000741C
CMI_COMPLETE    = 00000020
CMI_CTRL_CLEAR  = 00000001
CMI_CTRL_REG    = 0000741C
CMI_GO          = 00000008
CMI_READ        = 00000040
CMI_STATUS_0    = 00000008
CMI_STATUS_1    = 00000010
CMI_WRITE       = 00000020
CML             = 0000000E
COMPOSIT_LENGTH= 00000003
CON             = 00000011
CONTINUE_FLAG   = 00000002
CONTROL_C_FLAG  = 00000001
CONT_FLAG       = 00000020
CONT_KEY        = 0000001C
CPU_RUN         = 00000010
CSAMR           = 00007404
CSBUF           = 00007424
CSTRP           = 00007422
CS_ADD_BUFF_HGH= 00007425
CS_ADD_BUFF_LOW= 00007424
CS_ADD_TRAP_HGH= 00007423
CS_ADD_TRAP_LOW= 00007422
CYCLE_FLAG      = 00000008
CYCLE_KEY       = 00000020
D0              = 00000006
D1              = 00000003
D2              = 00000000
DATA_FLAG       = 00000000
DATA_LENGTH     = 00000001
DCSAD           = 00007401
DCSCF           = 00007403
DCSCR           = 00007402
DCSDA           = 00007400
DCS_ADDR_CLEAR  = 00000002
DCS_ADDR_REG_RO= 00007427
DCS_ADDR_REG_WO= 00007401
DCS_CTRL_FILE   = 00007403
DCS_CTRL_REG    = 00007402
DCS_DATA_REG    = 00007400
DCS_FLAG        = 00000000
DCS_FLAG1       = 00000000
DCS_HEAD        = 00000000 R      AA
DCS_LENGTH      = 00000001
DCS_SCRATCH_ADR= 00000036
DCS_START       = 00001800
DC_LOW          = 00000040
DIAGNOSE_FLAG   = 00000004
DPM             = 00000000
DUM1            = FFFFFFF6
ENDTEST_LOC     = 00000082 R      A9
END_SLICE       = 00000025
END_TEST_1        0000013D R      02
END_TEST_10       0000008A R      19
END_TEST_11       00000088 R      1B
END_TEST_12       000000CA R      1D
END_TEST_13       0000009C R      1F
END_TEST_14       000000A7 R      21
END_TEST_15       000000A4 R      23
END_TEST_16       000000A4 R      25
END_TEST_17       00000123 R      27
END_TEST_18       00000218 R      29
END_TEST_19       00000292 R      2B
END_TEST_2        0000007C R      09
END_TEST_20       000000F3 R      2D
END_TEST_21       000000F7 R      2F
END_TEST_22       00000097 R      31
END_TEST_23       000000A7 R      33
END_TEST_24       00000114 R      35
END_TEST_25       000000C0 R      37
END_TEST_26       000000C0 R      39
END_TEST_27       000000D9 R      3B
END_TEST_28       0000012A R      41
END_TEST_29       00000610 R      47
END_TEST_3        00000075 R      0B
END_TEST_30       000000A9 R      49
END_TEST_31       00000639 R      4B
END_TEST_32       000000D1 R      4D
END_TEST_33       0000017B R      53
END_TEST_34       00000130 R      55
END_TEST_35       00000104 R      57
END_TEST_36       00000125 R      59
END_TEST_37       000001C4 R      5B
END_TEST_38       0000014D R      5D
END_TEST_39       0000010C R      5F
END_TEST_4        0000007D R      0D
END_TEST_40       000000A1 R      61
END_TEST_41       00000078 R      63
END_TEST_42       000001B6 R      69
END_TEST_43       00000107 R      6F
END_TEST_44       0000023B R      75
END_TEST_45       00000269 R      7B
END_TEST_46       00000439 R      81
END_TEST_47       000002FE R      87
END_TEST_48       00000177 R      8D
END_TEST_49       00000374 R      8F
END_TEST_5        000000D4 R      0F
END_TEST_50       00000325 R      95
END_TEST_51       000001DC R      9B
END_TEST_52       0000006B R      9D
END_TEST_53       0000006B R      A3
END_TEST_54       0000007D R      A9
END_TEST_6        000004C1 R      11
END_TEST_7        000000A3 R      13
END_TEST_8        00000097 R      15
END_TEST_9        000000BF R      17
EN_TRACE_REG    = 00000008
ERRLOG_FLAG     = 00000000
ERROR_FLAG      = 00000010
ERROR_LOG_ADR   = 000032FC
ERR_LOG_INIT    = 00000001
EXP_STALL_FLAG  = 00000010
EXP_UTRAP_FLAG  = 00000040
FAC             = 00000020
FAULT           = 00000002
FCC             = 00000015
FCS             = 00000021
FETCH_FLAG      = 00000008
FEX             = 0000001E
FFH             = 00000024
FFL             = 00000023
FIC             = 0000001F
FIO             = 0000001D
FLAG_KEY        = 00000004
```

```
FMR             = 00000022          MAINT_TRAP_OFF  = 00000001          OP_LOAD_FIELD   = 00000026
FPA             = 00000002          MAINT_WB_TO_WH  = 00000008          OP_LOAD_INDEX   = 00000028
FP_BOOT_0       = 00000001          MAINT_WD_TO_WB  = 00000010          OP_LOAD_REG     = 00000016
FP_BOOT_1       = 00000002          MAP             = 00000012          OP_MASK         = 00000018
FP_START_0      = 00000004          MAREG           = 00007410          OP_NEW_TEST     = 0000001A
FP_START_1      = 00000008          MASTER_HALT_EN  = 00000008          OP_PATTERN_GEN  = 00000010
FQA             = 00000014          MAX_ARGUMENTS   = 00000010          OP_PRINT_N      = 00000036
FRONT1          = 00007420          MA_KEY          = 00000038          OP_PRINT_S      = 00000034
FRONT2          = 00007421          MC0             = 00000003          OP_SAVE_INDEX   = 0000002C
FRONT_PNL_1     = 00007420          MC1             = 00000009          OP_SKIP         = 0000001C
FRONT_PNL_2     = 00007421          MDL             = 0000001B          OP_SUB_TEST     = 0000001E
FRONT_PNL_LOCK  = 00000080          MDR             = 00000018          OVERLAY_HEAD    = 00000000 R     AE
HALT_FLAG       = 00000001          MDREG           = 00007414          PARITY_ERROR    = 00000020
HALT_KEY        = 00000008          MD_KEY          = 00000034          PAR_CHK_ENABL   = 00000008
HALT_ON_MATCH   = 00000001          MD_REG_BYTE_0   = 00007414          PAR_STOP_ENABL  = 00000010
HEAD            = FFFF7C00          MD_REG_BYTE_1   = 00007415          PASS_KEY        = 00000002
HIGH            = 00000001          MD_REG_BYTE_2   = 00007416          PB_INIT         = 00000040
H_FROM_WBUS     = 00000010          MD_REG_BYTE_3   = 00007417          PB_KEY          = 00000036
IB_FLAG         = 00000020          MEC             = 0000001A          PC_KEY          = 00000030
IB_KEY          = 00000012          MHREG           = 00007418          PHB             = 00000007
INSTR_FLAG      = 00000004          MH_REG_BYTE_0   = 00007418          PRK             = 0000000C
INSTR_KEY       = 00000018          MH_REG_BYTE_1   = 00007419          PS_KEY          = 0000003E
INT             = 00000010          MH_REG_BYTE_2   = 0000741A          QA_FLAG         = 00000040
IRD             = 00000005          MH_REG_BYTE_3   = 0000741B          QA_KEY          = 00000014
I_INDEX_FLAG    = 00000000          MIC             = 00000001          QV_FLAG         = 00000002
I_INIT          = 00000000          MICROWORD       = 0000000A          QV_KEY          = 00000028
J_INDEX_FLAG    = 00000000          MICRO_ADDR_INH  = 00000080          RDCTRL          = 0000741E
J_INIT          = 00000000          MONITOR_SIZE    = 00002C5E          RDM             = 00000004
K_INDEX_FLAG    = 00000000          MSQ             = 00000008          RDM_FLAG        = 00000004
K_INIT          = 00000000          MT_KEY          = 0000002E          RDM_KEY         = 0000002A
L               = 00000004          N               = 00000001          RDM_LAST        = 00000008
LIST_END        = 00000068 R     A9 NER_FLAG        = 00000004          RD_CTRL_REG     = 0000741E
LIST_HEAD       = 00000064 R     A9 NER_KEY         = 0000000C          REMOTE          = 00000001
LIST_LENGTH     = 00000004          NOERROR         = 00000001          RTEMP_FLAG      = 00000000
LITERAL         = 00000000          NOTEQUAL        = 00000003          RTEMP_FLAG1     = 00000000
LONG            = 00000004          ONEQUAL         = 00000002          RTEMP_HEAD      = 00000000 R     96
LOOP_ERROR_FLAG= 00000020           ONERROR         = 00000000          RTEMP_LENGTH    = 00000001
LOOP_FLAG       = 00000002          OP_BEGIN_LOOP   = 00000008          RT_KEY          = 0000002C
LOOP_KEY        = 0000000A          OP_BURST_CLOCK  = 00000002          SAC             = 00000009
LOST_FLAG       = 00000010          OP_COMPARE_REG  = 00000004          SA_CLOCK        = 00000080
LOST_KEY        = 0000001A          OP_COMPARE_REGM= 00000006           SA_FLAG         = 00000010
LOW             = 00000000          OP_COMPARE_VB   = 00000020          SA_KEY          = 00000010
M               = 0000000A          OP_DUMPLOG      = 00000030          SA_START_STOP   = 00000080
M$TEST_SIZE     = 000007A2          OP_ENABL_STALL  = 00000038          SBE_FLAG        = 00000020
M$TEST_SIZE_D   = 000003E2          OP_ENABL_TRAP   = 0000002E          SBE_KEY         = 00000042
MA0             = 00000008          OP_END_FILE     = 00000024          SECTION_FLAG    = 00000080
MA1             = 0000000A          OP_END_LOOP     = 0000000A          SF_KEY          = 00000040
MAD             = 00000013          OP_END_TEST     = 0000000C          SOMM_FLAG       = 00000001
MAIN32          = 0000741D          OP_ERRLOG       = 00000032          SOMM_KEY        = 00000006
MAINT_A_TO_CMI  = 00000080          OP_ERROR_LOOP   = 0000000E          SPA             = 00000002
MAINT_CMI_TO_MH= 00000020           OP_FETCH        = 00000022          SRK             = 00000001
MAINT_DCS_ENABL= 00000004           OP_IF_ERROR     = 00000012          SRM             = 00000017
MAINT_MD_TO_CMI= 00000040           OP_INC_INDEX    = 0000002A          SR_KEY          = 0000003C
MAINT_REG       = 0000741D          OP_INIT         = 00000014          STACK_SIZE      = 00000400
MAINT_STB_INH   = 00000002          OP_LOAD_DCS     = 00000000          STATUS          = 00007406
```

```
STATUS_REG      = 00007406
STEP_KEY        = 0000001E
STOP_CLOCK      = 00000004
STROBE_CMI      = 00000040
ST_KEY          = 0000001E
TEMP            = 00000002 R      A8
TEMP1           = 00000044
TEMP2           = 0000003A
TEMP3           = 00000002
TEMP4           = 00000002 R      A8
TEST            = 00000004
TEST_FLAG       = 00000000
TEST_HEAD       = 0000002F R      A9
TEST_KEY        = 00000000
TEST_LENGTH     = 00000002
TEST_SECT_HEAD  = 00000000 R      A9
THIS_TEST       = 00000036
TICK_FLAG       = 00000002
TICK_KEY        = 00000022
TOK             = 00000006
TRACE_ENABL     = 00000004
TRAP_HALT_EN    = 00000020
TRAP_OFF        = 00000002
TR_FLAG         = 00000080
TR_KEY          = 00000016
TSTSPAN_FLAG    = 00000040
UBI             = 00000006
UDP             = 0000001C
UTR             = 0000000D
VA_KEY          = 00000032
VBUS_CLOCK      = 00000080
VBUS_COMPARE    = 00000080
VBUS_LOAD       = 00000010
VBUS_SERIAL_IN  = 00000040
VBUS_SERIAL_OUT = 00000001
VB_KEY          = 00000026
V_BUS_STROBE    = 00000001
W               = 00000002
WBUS_FROM_D     = 00000020
WDREG           = 0000742C
WD_KEY          = 0000003A
WD_REG_BYTE_0   = 0000742C
WD_REG_BYTE_1   = 0000742D
WD_REG_BYTE_2   = 0000742E
WD_REG_BYTE_3   = 0000742F
WHREG           = 00007430
WH_REG_BYTE_0   = 00007430
WH_REG_BYTE_1   = 00007431
WH_REG_BYTE_2   = 00007432
WH_REG_BYTE_3   = 00007433
WORD            = 00000002
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
X_0001_D	00000002 (2.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0001_T	00000180 (384.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0001_GDCS	000001A3 (419.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0001_ERTEMP	00000021 (33.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0001_FCACHE	00000001 (1.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0001_HFILL	00000039 (57.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
DIRECTORY	00000036 (54.)	07 (7.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0002_D	00000002 (2.)	08 (8.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0002_T	000000FE (254.)	09 (9.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0003_D	00000002 (2.)	0A (10.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0003_T	0000007E (126.)	0B (11.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0004_D	00000002 (2.)	0C (12.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0004_T	000000FE (254.)	0D (13.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0005_D	00000002 (2.)	0E (14.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0005_T	000000FE (254.)	0F (15.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0006_D	00000002 (2.)	10 (16.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0006_T	000004FE (1278.)	11 (17.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0007_D	00000002 (2.)	12 (18.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0007_T	000000FE (254.)	13 (19.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0008_D	00000002 (2.)	14 (20.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0008_T	000000FE (254.)	15 (21.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0009_D	00000002 (2.)	16 (22.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0009_T	000000FE (254.)	17 (23.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0010_D	00000002 (2.)	18 (24.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0010_T	000000FE (254.)	19 (25.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0011_D	00000002 (2.)	1A (26.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0011_T	000000FE (254.)	1B (27.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0012_D	00000002 (2.)	1C (28.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0012_T	000000FE (254.)	1D (29.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0013_D	00000002 (2.)	1E (30.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0013_T	000000FE (254.)	1F (31.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0014_D	00000002 (2.)	20 (32.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0014_T	000000FE (254.)	21 (33.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0015_D	00000002 (2.)	22 (34.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0015_T	000000FE (254.)	23 (35.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0016_D	00000002 (2.)	24 (36.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0016_T	000000FE (254.)	25 (37.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0017_D	00000002 (2.)	26 (38.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0017_T	0000017E (382.)	27 (39.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0018_D	00000002 (2.)	28 (40.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0018_T	0000027E (638.)	29 (41.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0019_D	00000002 (2.)	2A (42.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0019_T	000002FE (766.)	2B (43.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0020_D	00000002 (2.)	2C (44.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0020_T	000000FE (254.)	2D (45.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0021_D	00000002 (2.)	2E (46.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0021_T	000000FE (254.)	2F (47.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

X_0022_D	00000002	(2.)	30	(48.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0022_T	000000FE	(254.)	31	(49.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0023_D	00000002	(2.)	32	(50.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0023_T	000000FE	(254.)	33	(51.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0024_D	00000002	(2.)	34	(52.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0024_T	0000017E	(382.)	35	(53.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0025_D	00000002	(2.)	36	(54.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0025_T	000000FE	(254.)	37	(55.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0026_D	00000002	(2.)	38	(56.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0026_T	000000FE	(254.)	39	(57.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0027_D	00000002	(2.)	3A	(58.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0027_T	00000100	(256.)	3B	(59.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0027_ERTEMP	00000015	(21.)	3C	(60.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0027_FCACHE	00000001	(1.)	3D	(61.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0027_GDCS	00000001	(1.)	3E	(62.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0027_HFILL	00000067	(103.)	3F	(63.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0028_D	00000002	(2.)	40	(64.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0028_T	00000180	(384.)	41	(65.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0028_ERTEMP	00000021	(33.)	42	(66.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0028_FCACHE	00000001	(1.)	43	(67.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0028_GDCS	00000001	(1.)	44	(68.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0028_HFILL	0000005B	(91.)	45	(69.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0029_D	00000002	(2.)	46	(70.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0029_T	0000067E	(1662.)	47	(71.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0030_D	00000002	(2.)	48	(72.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0030_T	000000FE	(254.)	49	(73.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0031_D	00000002	(2.)	4A	(74.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0031_T	0000067E	(1662.)	4B	(75.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0032_D	00000002	(2.)	4C	(76.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0032_T	00000100	(256.)	4D	(77.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0032_GDCS	00000059	(89.)	4E	(78.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0032_ERTEMP	00000001	(1.)	4F	(79.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0032_FCACHE	00000001	(1.)	50	(80.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0032_HFILL	00000023	(35.)	51	(81.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0033_D	00000002	(2.)	52	(82.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0033_T	000001FE	(510.)	53	(83.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0034_D	00000002	(2.)	54	(84.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0034_T	0000017E	(382.)	55	(85.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0035_D	00000002	(2.)	56	(86.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0035_T	0000017E	(382.)	57	(87.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0036_D	00000002	(2.)	58	(88.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0036_T	0000017E	(382.)	59	(89.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0037_D	00000002	(2.)	5A	(90.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0037_T	000001FE	(510.)	5B	(91.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0038_D	00000002	(2.)	5C	(92.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0038_T	0000017E	(382.)	5D	(93.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0039_D	00000002	(2.)	5E	(94.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0039_T	0000017E	(382.)	5F	(95.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0040_D	00000002	(2.)	60	(96.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0040_T	000000FE	(254.)	61	(97.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0041_D	00000002	(2.)	62	(98.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0041_T	00000080	(128.)	63	(99.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0041_GDCS	0000002D	(45.)	64	(100.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0041_ERTEMP	00000001	(1.)	65	(101.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0041_FCACHE	00000001	(1.)	66	(102.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE

X_0041_HFILL	0000004F	(79.)	67 (103.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0042_D	00000002	(2.)	68 (104.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0042_T	00000200	(512.)	69 (105.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0042_GDCS	00000038	(56.)	6A (106.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0042_ERTEMP	00000009	(9.)	6B (107.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0042_FCACHE	00000001	(1.)	6C (108.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0042_HFILL	0000003C	(60.)	6D (109.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0043_D	00000002	(2.)	6E (110.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0043_T	00000180	(384.)	6F (111.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0043_GDCS	00000038	(56.)	70 (112.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0043_ERTEMP	0000000D	(13.)	71 (113.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0043_FCACHE	00000001	(1.)	72 (114.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0043_HFILL	00000038	(56.)	73 (115.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0044_D	00000002	(2.)	74 (116.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0044_T	00000280	(640.)	75 (117.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0044_ERTEMP	00000021	(33.)	76 (118.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0044_FCACHE	00000001	(1.)	77 (119.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0044_GDCS	00000001	(1.)	78 (120.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0044_HFILL	0000005B	(91.)	79 (121.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0045_D	00000002	(2.)	7A (122.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0045_T	00000280	(640.)	7B (123.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0045_ERTEMP	00000025	(37.)	7C (124.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0045_FCACHE	00000001	(1.)	7D (125.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0045_GDCS	00000001	(1.)	7E (126.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0045_HFILL	00000057	(87.)	7F (127.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0046_D	00000002	(2.)	80 (128.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0046_T	00000480	(1152.)	81 (129.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0046_ERTEMP	00000031	(49.)	82 (130.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0046_FCACHE	00000001	(1.)	83 (131.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0046_GDCS	00000001	(1.)	84 (132.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0046_HFILL	0000004B	(75.)	85 (133.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0047_D	00000002	(2.)	86 (134.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0047_T	00000380	(896.)	87 (135.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0047_ERTEMP	0000001D	(29.)	88 (136.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0047_FCACHE	00000001	(1.)	89 (137.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0047_GDCS	00000001	(1.)	8A (138.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0047_HFILL	0000005F	(95.)	8B (139.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0048_D	00000002	(2.)	8C (140.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0048_T	0000017E	(382.)	8D (141.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0049_D	00000002	(2.)	8E (142.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0049_T	00000380	(896.)	8F (143.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0049_ERTEMP	00000031	(49.)	90 (144.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0049_FCACHE	00000001	(1.)	91 (145.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0049_GDCS	00000001	(1.)	92 (146.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0049_HFILL	0000004B	(75.)	93 (147.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0050_D	00000002	(2.)	94 (148.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0050_T	00000380	(896.)	95 (149.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0050_ERTEMP	00000019	(25.)	96 (150.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0050_FCACHE	00000001	(1.)	97 (151.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0050_GDCS	00000001	(1.)	98 (152.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0050_HFILL	00000063	(99.)	99 (153.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0051_D	00000002	(2.)	9A (154.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0051_T	000001FE	(510.)	9B (155.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0052_D	00000002	(2.)	9C (156.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0052_T	00000080	(128.)	9D (157.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE

```
X_0052_GDCS                     0000002D  (   45.)  9E (158.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0052_ERTEMP                   00000001  (    1.)  9F (159.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0052_FCACHE                   00000001  (    1.)  A0 (160.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0052_HFILL                    0000004F  (   79.)  A1 (161.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0053_D                        00000002  (    2.)  A2 (162.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0053_T                        00000080  (  128.)  A3 (163.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0053_GDCS                     0000002D  (   45.)  A4 (164.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0053_ERTEMP                   00000001  (    1.)  A5 (165.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0053_FCACHE                   00000001  (    1.)  A6 (166.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0053_HFILL                    0000004F  (   79.)  A7 (167.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0054_D                        00000002  (    2.)  A8 (168.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0054_T                        00000100  (  256.)  A9 (169.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0054_GDCS                     00000038  (   56.)  AA (170.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0054_ERTEMP                   00000001  (    1.)  AB (171.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0054_FCACHE                   00000001  (    1.)  AC (172.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0054_HFILL                    00000044  (   68.)  AD (173.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0055_D                        00000000  (    0.)  AE (174.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
```

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	28	00:00:00.12	00:00:01.15
Command processing	861	00:00:00.71	00:00:02.41
Pass 1	7067	00:02:30.08	00:06:49.01
Symbol table sort	1	00:00:00.56	00:00:00.74
Pass 2	6930	00:00:42.94	00:01:38.88
Symbol table output	2	00:00:00.18	00:00:00.79
Psect synopsis output	3	00:00:00.40	00:00:00.62
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	14895	00:03:14.99	00:08:33.63

The working set limit was 4096 pages.
2848330 bytes (5564 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 381 non-local and 418 local symbols.
11586 source lines were read in Pass 1, producing 399 object records in Pass 2.
99 pages of virtual memory were used to define 45 macros.

Macro library statistics

Macro library name	Macros defined
DISK$UCODPACK00:[VAX750]COMETMAC.MLB;805	45
SYS$COMMON:[SYSLIB]STARLET.MLB;2	0
TOTALS (all libraries)	45

1906 GETS were required to define 45 macros.

There were 0 errors, 4 warnings and 0 information messages, on lines:
 1223 (1) 4661 (1) 5528 (1) 9022 (1)

```
ZZ-ECKAB-9.2    VAX-11 Macro Run Statistics                    13-MAY-1987       Fiche 2 Frame G5          Sequence 264
ECKAB                          VAX-11/750 DPM MICRO DIAGNOSTICS    11-DEC-1985 09:32:54  VAX/VMS Macro V04-00        Page 23
VAX-11 Macro Run Statistics                                        11-DEC-1985 09:32:27  [VAX750.DPM]ECKAB1.MAR;1          (1

MACRO UCOD$0:[VAX750.DPM]TITLE+DISK$UCODPACK00:[VAX750]COMETASM+UCOD$0:[VAX750.DPM]ECKAB1/LIS=ECKAB1/OBJ=ECKAB1
```

(1) 24 "DPM Revision History"
(1) 1 ""
(1) 3 "DIAGNOSTIC MICROCODE MACROS"
(1) 98 " RDM Control File Macros"
(1) 109 " Diagnostic Macros"
(1) 194 ""
(1) 195 "MODULE GATE ARRAY MAPS"
(1) 197 " L0001 MULTIPLIER AND GATE ARRAY LOCATION"
(1) 254 " L0002 GATE ARRAY LOCATION"
(1) 311 " L0003 GATE ARRAY LOCATION"
(1) 369 " L0004 GATE ARRAY LOCATION"
(1) 427 " L0011 & L0016 GATE ARRAY LOCATION"
(1) 485 ""
(1) 490 "EQUATED SYMBOLS
(1) 493 ""
(1) 2 "TEST 037 Rotator Test 5 PLATCH writes to SBUS when SLATCH gets WBUS Test
(1) 116 "TEST 038 Rotator Test 6 SLATCH writes to SBUS when PLATCH gets WBUS Test
(1) 230 "TEST 039 Rotator Test 7 PLATCH gets Literal Test
(1) 342 "TEST 03A Rotator Test 8 SLATCH gets Literal Test
(1) 454 "TEST 03B Rotator Test 9 Zero Test
(1) 577 "TEST 03C Rotator Test 10 Minus One Test
(1) 701 "TEST 03D Rotator Test 11 DSIZE Constants Test
(1) 858 "TEST 03E Rotator Test 12 Extract MBUS concatenated with RBUS Test 1
(1) 995 "TEST 03F Rotator Test 13 Extract MBUS concatenated with RBUS Test 2
(1) 1167 "TEST 040 Rotator Test 14 Extract MBUS concatenated with MBUS Test
(1) 1346 "TEST 041 Rotator Test 15 Extract RBUS concatenated with RBUS Test
(1) 1524 "TEST 042 Rotator Test 16 Extract PTE Address Test
(1) 1640 "TEST 043 Rotator Test 17 Extract Virtual Page Number Test
(1) 1757 "TEST 044 Rotator Test 18 Nibble Extract Test
(1) 1869 "TEST 045 Rotator Test 19 Exponent Extract Test
(1) 1986 "TEST 046 Rotator Test 20 Clear MBUS Bytes Test
(1) 2171 "TEST 047 Rotator Test 21 Zero Extended Literal Test
(1) 2314 "TEST 048 Rotator Test 22 One Extended Literal Test
(1) 2495 "TEST 049 Rotator Test 23 Right Rotation of RBUS Test 1
(1) 2821 "TEST 04A Rotator Test 24 Right Rotation of RBUS Test 2
(1) 3111 "TEST 04B Rotator Test 25 Right Rotation of MBUS Test 1
(1) 3432 "TEST 04C Rotator Test 26 Right Rotation of MBUS Test 2
(1) 3722 "TEST 04D Rotator Test 27 Left Rotation of RBUS Test
(1) 4050 "TEST 04E Rotator Test 28 Left Rotation of MBUS Test
(1) 4418 "TEST 04F Rotator Test 29 Right Rotation of MBUS concat. with RBUS Test 1
(1) 4808 "TEST 050 Rotator Test 30 Right Rotation of MBUS concat. with RBUS Test 2
(1) 5091 "TEST 051 Rotator Test 31 Right Rotation of MBUS concat. with RBUS Test 3
(1) 5324 "TEST 052 Rotator Test 32 Left Rotation of RBUS concat. with MBUS Test 1
(1) 5604 "TEST 053 Rotator Test 33 Left Rotation of RBUS concat. with MBUS Test 2
(1) 5987 "TEST 054 Rotator Test 34 Zero Extend and Rotate Left Literal Test
(1) 6134 "TEST 055 Rotator Test 35 One Extend and Rotate Left Literal Test
(1) 6269 "TEST 056 Rotator Test 36 Arithmetic Right Shift of MBUS by PLATCH Test
(1) 6496 "TEST 057 Rotator Test 37 Arithmetic Right Shift of MBUS by -PLATCH Test
(1) 6651 "TEST 058 Rotator Test 38 Arithmetic Right Shift of MBUS by 3 Test
(1) 6741 "TEST 059 Rotator Test 39 Arithmetic Left Shift of MBUS by PLATCH Test
(1) 6872 "TEST 05A Rotator Test 40 Arithmetic Left Shift of RBUS by PLATCH Test
(1) 7003 "TEST 05B Rotator Test 41 Arithmetic Left Shift of RBUS by Size Test
(1) 7170 "TEST 05C Rotator Test 42 Arithmetic Left Shift of RBUS by 7 Test
(1) 7276 "TEST 05D Rotator Test 43 Unpack Floating Point Fraction Test
(1) 7467 "TEST 05E Rotator Test 44 Pack Floating Point Number Test
(1) 7658 "TEST 05F Rotator Test 45 Floating Point Literal Test
(1) 7794 "TEST 060 Rotator Test 46 Find Most Significant Bit Test

```
    (1)     7957        "TEST   061     Rotator Test 47 BCD Swap Test
    (1)     8087        "TEST   062     Rotator Test 48 Convert Numeric to BCD Test
```

```
0000     1          .TITLE  ECKAB VAX-11/750 DPM MICRO DIAGNOSTICS
0000     2          .IDENT  /V09.01/
0000     3 ;
0000     4 ; COPYRIGHT (C) 1980,1983
0000     5 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     6 ;
0000     7 ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE  ONLY ON A SINGLE
0000     8 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000     9 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    10 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    11 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    12 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    13 ; REMAIN IN DEC.
0000    14 ;
0000    15 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    16 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    17 ; CORPORATION.
0000    18 ;
0000    19 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    20 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    21 ;
0000    22 ;
```

```
0000     24          .SBTTL  "DPM Revision History"
0000     25 ;++
0000     26 ; DPM MICRO-DIAGNOSTIC PACKAGE
0000     27 ;
0000     28 ; REVISION HISTORY
0000     29 ;
0000     30 ; 00-01 Tom Groetzinger 4-FEB-1980
0000     31 ;       Started file.
0000     32 ;
0000     33 ; 00-02 Dave Spain      27-FEB-1980
0000     34 ;       Added CS BUS INTEGRITY test.
0000     35 ;
0000     36 ; 00-03 Tom Groetzinger 12-MAR-1980
0000     37 ;       Added five tests.
0000     38 ;
0000     39 ; 00-04 Tom Groetzinger 17-MAR-1980
0000     40 ;       Added SRK,SRM callouts to some tests.
0000     41 ;
0000     42 ; 00-05 Tom Groetzinger 17-MAR-1980
0000     43 ;       Improved tests 7 and 16 to increase coverage due to Westboro feedback
0000     44 ;
0000     45 ; 00-06 Tom Groetzinger 17-MAR-1980
0000     46 ;       Debugged test 16 dueu to major rewrite
0000     47 ;
0000     48 ; 00-07 Tom Groetzinger 17-MAR-1980
0000     49 ;       Added 9 Super Rotator tests
0000     50 ;
0000     51 ; 01-08 Dave Spain       4-APR-80
0000     52 ;       Added 4 Micro-sequencer tests for offical release.
0000     53 ;
0000     54 ; 01-09 Bill Landry     4-APR-1980
0000     55 ;       Added the XB, PREFETCH and IRD tests from ECKAC.SRC so
0000     56 ;       manufacturing can run one tape for DPM.
0000     57 ;
0000     58 ; 00-10 Bill Landry     4-APR-1980
0000     59 ;       Let's save version 1.0 for our first SDC release.
0000     60 ;
0000     61 ; 00-11 Tom Groetzinger 7-APR-1980
0000     62 ;       Added 2 ALK Tests
0000     63 ;
0000     64 ; 00-12 Tom Groetzinger 8-APR-1980
0000     65 ;       Deleted last 2 D Register outputs from ALK Test #6
0000     66 ;
0000     67 ; 00-13 Tom Groetzinger 14-APR-1980
0000     68 ;       Added D Register output to ALK test #6
0000     69 ;
0000     70 ; 00-14 Dave Spain      25-APR-1980
0000     71 ;       Changed mask on the micro-sequencer tests to ignore bit 13 of the
0000     72 ;       Control Store address.
0000     73 ;
0000     74 ; 00-15 Dave Spain      28-APR-1980
0000     75 ;       Fixed bug in ALK Tests 6 and 7 by replacing VSIZE/1 micro order with a
0000     76 ;       SIZE[LONG] macro in micro words 6 and 3 respectively.
0000     77 ;
0000     78 ; 00-16 Dave Spain      29-APR-1980
```

```
0000     79 ;       Changed mask on CS address for Instruction Address Decode test from
0000     80 ;       3FFF to 1FFF to ignore bit 13 of Control Store address.
0000     81 ;
0000     82 ; 00-17 Dave Spain      30-APR-1980
0000     83 ;       Added CCS module to error callout for Control Store Bus Integrity test.
0000     84 ;
0000     85 ; 00-18 Dave Spain       9-MAY-1980
0000     86 ;       Added SRM gate array to error callout routine in RTMP0 and RTEMP
0000     87 ;       Sratchpad tests.
0000     88 ;
0000     89 ; 00-19 Bill Landry     10-JUN-1980
0000     90 ;       Added RDM tests to beginning of tape
0000     91 ;
0000     92 ; 01-00 Bill Landry/Dave Spain  13-JUN-1980
0000     93 ;       Added macro file to title section and compatibility mode IRD tests
0000     94 ;       then submitted to RELEASE ENGINERRING.
0000     95 ;
0000     96 ; 01-01 Dave Spain      19-JUN-1980
0000     97 ;       Fixed prefetch incrementer test to clear out cache data after every
0000     98 ;       prefetch. This will prevent the test from mistaking bad addresses from
0000     99 ;       good ones.
0000    100 ;
0000    101 ; 01-02 Dave Spain       2-JUL-1980
0000    102 ;       Added CM Second IRDX tests to file.
0000    103 ;
0000    104 ; 01-03 Dave Spain      18-AUG-1980
0000    105 ;       Added CM branch test to file.
0000    106 ;
0000    107 ; 01-04 Dave Spain      10-SEPT-1980
0000    108 ;       Fixed "DEFAULT MTEMP0 TEST"
0000    109 ;
0000    110 ; 01-05 Bill Landry     11-SEPT-1980
0000    111 ;       Revised RDM to CMI tests
0000    112 ;
0000    113 ; 02-00 Dave Spain      28-SEPT-1980
0000    114 ;       Added rotator tests to file and submitted to SDC.
0000    115 ;
0000    116 ; 02-01 Dave Spain      28-OCT-1980
0000    117 ;       Added QV flag to tests to be run by manufacturing.
0000    118 ;
0000    119 ; 02-02 Bill Landry     10-NOV-1980
0000    120 ;       Added tests for TOK and SAC gate arrays.
0000    121 ;
0000    122 ; 02-03 Bill Landry     13-NOV-1980
0000    123 ;       Added test of REI CHECK
0000    124 ;
0000    125 ; 03-00 Dave Spain      17-NOV-1980
0000    126 ;       Added rotator tests 23-28 and released to SDC.
0000    127 ;
0000    128 ; 03-01 Dave Spain      10-DEC-1980
0000    129 ;       Added rotator tests 29-42.
0000    130 ;
0000    131 ; 03-02 Bill Landry     11-DEC-1980
0000    132 ;       Fixed bug in CS ADDRESS test
0000    133 ;
```

```
0000   134 ; 03-03 Dave Spain      12-DEC-1980
0000   135 ;       Enhanced Rotator Test 1 to call out ALP slices.
0000   136 ;
0000   137 ; 03-04 Bill Landry     16-DEC-1980
0000   138 ;       Added DSIZE ROM tests
0000   139 ;
0000   140 ; 03-05 Bill Landry     31-DEC-1980
0000   141 ;       Last great act of 80. Separated DPM and RDM tests into two files.
0000   142 ;
0000   143 ; 04-00 Bill Landry     5-JAN-1981
0000   144 ;       Submitted to RELEASE ENGINERRING
0000   145 ;
0000   146 ; 04-01 Bill Landry     14-JAN-1981
0000   147 ;       Revised INTERVAL TIMER INTERRUPT TEST to fix bug caused by INT PEND
0000   148 ;       ECO
0000   149 ;
0000   150 ; 04-02 Dave Spain      16-JAN-1981
0000   151 ;       Fixed error callout in CS Parity Error, XB, and Prefetch Incrementer
0000   152 ;       tests and added seven new rotator tests.
0000   153 ;
0000   154 ; 04-03 Bill Landry     4-FEB-1981
0000   155 ;       Added tests for CCC gate array
0000   156 ;
0000   157 ; 04-04 Dave Spain      9-FEB-1981
0000   158 ;       Changed the names of rotator tests 23-33.
0000   159 ;
0000   160 ; 04-05 Bill Landry     24-FEB-1981
0000   161 ;       Added tests for DPM discrete branching logic.
0000   162 ;
0000   163 ; 04-06 Bill Landry     27-FEB-1981
0000   164 ;       Added tests for branching logic in PHB gate array.
0000   165 ;
0000   166 ; 05-00 Dave Spain       1-MAR-1981
0000   167 ;       Added 5 SRK status rotator tests and submitted to SDC.
0000   168 ;
0000   169 ; 05-01 Dave Spain      25-MAR-1981
0000   170 ;       Fixed a loop on test bug in the Alu Functions Test by adding a LDFIELD
0000   171 ;       pseudo-op that reinitializes the ROT field in Subtest 1.
0000   172 ;
0000   173 ; 05-02 Bill Landry     25-MAR-1981
0000   174 ;       Fixed bug in BUT/NO.FPA test.  I was branching on the wrong conditions.
0000   175 ;
0000   176 ; 05-03 Dave Spain      14-APR-1981
0000   177 ;       Cleaned up documentation for R Scratchpads Write All Cells Test.
0000   178 ;
0000   179 ; 06-00 Bill Landry     2-MAY-1981
0000   180 ;       Added SPA branching, CON registers, TU58 registers, and CCS PARITY
0000   181 ;       tests.  Submitted to Release Engineering.
0000   182 ;
0000   183 ; 06-01 Dave Spain      17-JUN-1981
0000   184 ;       Replaced the old IRD1 Instruction Address Decode test with two new
0000   185 ;       tests that toggle the PSL's FPD bit and can handle an FPA.
0000   186 ;
0000   187 ; 06-02 Bill Landry     19-AUG-1981
0000   188 ;       Fixed bugs created by introduction of FPA option.
```

```
0000    189 ;       Replaced all references to COMET with the word CPU.
0000    190 ;       Fixed typing error in vbus data of CPU CLOCK TEST.
0000    191 ;       Submitted to RELEASE ENGINEERING.
0000    192 ;
0000    193 ; 06-03 Bill Landry     24-SEPT-1981
0000    194 ;       Fixed potential clock problem with FPA in Z BIT TEST 1.
0000    195 ;
0000    196 ; 06-04 Bill Landry     9-OCT-1981
0000    197 ;       Added changes for ECO L0002-TW002
0000    198 ;
0000    199 ; 07-00 Bill Landry     31-OCT-1981
0000    200 ;       Removed changes for ECO L0002-TW002.  ECO is on hold.
0000    201 ;       Added tests for FPA to CPU interface logic
0000    202 ;       Moved WCS test to MIC tape
0000    203 ;       Submitted to RELEASE ENGINEERING.
0000    204 ;
0000    205 ; 07-01 Bill Landry     13-NOV-1981
0000    206 ;       Stopped release of 7.0
0000    207 ;       Added FPA TRAP LOGIC TEST
0000    208 ;       Enhanced error messages for FPA
0000    209 ;       Resubmitted to RELEASE ENGINEERING
0000    210 ;
0000    211 ; 07-02 Bill Landry     20-NOV-1981
0000    212 ;       Stopped release of 7.1
0000    213 ;       Enhanced FPA ENABLE/DISABLE TEST to check for MSEQ INIT
0000    214 ;       Resubmitted to RELEASE ENGINEERING
0000    215 ;
0000    216 ; 07-03 Dave Shull      26-Mar-1982
0000    217 ;       Added the ERRLOOP pseudo to the Rotator Extract tests which were
0000    218 ;       missing these and couldn't be looped on.
0000    219 ;
0000    220 ; 07-04 Dave Shull      05-Apr-1982
0000    221 ;       Removed all BEGINSA and ENDSA pseudo's.
0000    222 ;       Submitted to RELEASE ENGINEERING
0000    223 ;
0000    224 ; 07-05 Dave Shull      21-May-1982
0000    225 ;       Add a subtest to the Micro Sequencer Stack Test that will test bits
0000    226 ;       11 and 12 of the micro stack.
0000    227 ;
0000    228 ; 07-06 Dave Shull      24-May-1982
0000    229 ;       Changed documentation in the CM Odd Address BUT Test to reflect the
0000    230 ;       ECO L0002-TW002
0000    231 ;
0000    232 ; 08-00 Dave Shull      18-Jun-1982
0000    233 ;       Added 10 new tests.  Two of those tests are an addition to the existing
0000    234 ;       IRD1 rom tests.  These 2 new tests check the signal ROM OS INH H.
0000    235 ;       The other 8 new tests check the validity of the IRDX ROMs.  They check
0000    236 ;       both the address and the signal ROM OS INH H.
0000    237 ;
0000    238 ; 08-01 Dave Shull      22-Jun-1982
0000    239 ;       Added a subtest to the Register Backup Stack test that will check the
0000    240 ;       bit that records Increment/Decrement of the GPR.
0000    241 ;
0000    242 ; 09-00 Dave Shull      03-Nov-1982
0000    243 ;       Added two new Tests for coverage the new PCS (L0008) module.
```

```
0000    244 ;
0000    245 ; 09-01 Dave Smith      15-May-1983
0000    246 ;       Correct DCS in ALK test 3,6,7. Disabled ALP's on the WBUS.
0000    247 ;--
0000      1         .SBTTL  ""
```

```
0000     3          .SBTTL  "DIAGNOSTIC MICROCODE MACROS"
0000     4 ;++
0000     5 ; Revision History
0000     6 ;
0000     7 ; 28    Dave Spain      22-Jun-82
0000     8 ;       Removed LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
0000     9 ;       Decided not to use them.
0000    10 ;
0000    11 ; 27    Dave Spain      15-Jun-82
0000    12 ;       Added LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
0000    13 ;
0000    14 ; 26    Dave Spain      19-May-82
0000    15 ;       Added M[]_FPA macro.
0000    16 ;
0000    17 ; 25    Rich Lewis      23-Feb-82
0000    18 ;       Added PSL_WB, M[]_R[].OR.NIB0(M), WB_FPA CCR, WB_FPA TCR, Q_R[].ASL.P,
0000    19 ;       RDM_M[].XZ.OR.R[], FPA_WB macros
0000    20 ;
0000    21 ; 24    Dave Spain      02-Feb-82
0000    22 ;       Added FPA_R[]+M[] macro.
0000    23 ;
0000    24 ; 23    Bill Landry     22-OCT-81
0000    25 ;       Added PSL_RDM macro
0000    26 ;
0000    27 ; 22    Dave Spain       5 Oct 81
0000    28 ;       Added RDM_FPA and FPA_M[] FPA_RDM macros.
0000    29 ;
0000    30 ; 21    Bill Landry     29 Sept 81
0000    31 ;       Added FPA macro, FPA_M[] FPA_R[].
0000    32 ;
0000    33 ; 20    Dave Spain      14 Aug 81
0000    34 ;       Added the TESTFPA [] macro.
0000    35 ;
0000    36 ; 19    Dave Spain      18 July 81
0000    37 ;       Added the following FPA macros, FPA_RDM and FPA_R[].
0000    38 ;
0000    39 ; 18    Dave Spain      24 Sept 80
0000    40 ;       Added RDM_Q macro. This macro uses the ALU.
0000    41 ;
0000    42 ; 17    Bill Landry     23 Sept 80
0000    43 ;       Added macro VA_VA-ZLIT0[] for testing memory in decending order.
0000    44 ;
0000    45 ; 16    Dave Spain      27 August 80
0000    46 ;       Changed RDM_Q macro to RDM_Q Q_D. This more accurately reflects macro.
0000    47 ;
0000    48 ; 15    Bill Landry     4 June 80
0000    49 ;       Deleted BUS/PRB.RD.PTE from macro CLRTB.VA_VA and CLRCH.VA_VA to
0000    50 ;       satisfy validity checks.
0000    51 ;
0000    52 ; 14    Dave Spain      23 May 80
0000    53 ;       Added BUS/PRB.RD.PTE to the clear cache and clear TB macros for Mr.
0000    54 ;       Bill.
0000    55 ;
0000    56 ; 13    Dave Spain      14 Apr 80
0000    57 ;       Added SIZE[] or VSIZE/1 to some macros. This allows version 65 of
```

```
0000     58 ;        the COMET micro-code defines to run.
0000     59 ;
0000     60 ; 12     Bill Landry       1 Apr 80
0000     61 ;        Added BUS/PRB.RD micro order to RDM_TB macro to compensate for timing
0000     62 ;        problem in ADD gate array.  This is not an APRIL FOOL!
0000     63 ;
0000     64 ; 11     Dave Spain       27 Mar 80
0000     65 ;        Fixed TB macro's to include MSRC/VA to prevent validity failure.
0000     66 ;
0000     67 ; 10     Bill Landry      19 Mar 80
0000     68 ;        Added Q_R[].RL.P and VA_Q.OR.R[]
0000     69 ;
0000     70 ; 09     Bill Landry      19 Mar 80
0000     71 ;        Deleted TB_R[].RL.P.OR.D
0000     72 ;
0000     73 ; 08     Bill Landry      18 Mar 80
0000     74 ;        Added VA_R[].RL.P
0000     75 ;
0000     76 ; 07     Bill Landry      07 Mar 80
0000     77 ;        Added Q_M[].RL.PTE
0000     78 ;
0000     79 ; 06     Dave Spain       06 Mar 80
0000     80 ;        Changed TB_R[].RL.P macro to TB_R[].RL.P.OR.D.
0000     81 ;
0000     82 ; 05     Dave Spain       03 Mar 80
0000     83 ;        Added TB_R[].RL.P macro for Bill Landry.
0000     84 ;
0000     85 ; 04     Dave Spain       22 Feb 80
0000     86 ;        Added CLRTB.VA_R[] macro for Bill Landry.
0000     87 ;
0000     88 ; 03     Dave Spain       30 Jan 80
0000     89 ;        Added Tom Groetzinger's macro definitions.
0000     90 ;
0000     91 ; 02     Dave Spain       30 Jan 80
0000     92 ;        Added Tony Vezza's macro definitions.
0000     93 ;
0000     94 ; 01     Dave Spain       29 Jan 80
0000     95 ;        Macros initiated.
0000     96 ;--
0000     97
0000     98          .SBTTL  "       RDM Control File Macros"
0000     99
0000    100 ;STROBE.V.BUS            "RDMCF0/V.STROBE"
0000    101 ;DCS.ADDR_0              "RDMCF1/DCS.ADDR.CLR"
0000    102 ;STOP.CLOCK              "RDMCF2/STOP.CPU.CLOCK"
0000    103 ;LATCH.CS.ADDR           "RDMCF3/DIS.STROBE.CS.A"
0000    104 ;RDM_WB                  "RDMCF4/H.REG.FROM.WB"
0000    105 ;WB_RDM                  "RDMCF5/WB.FROM.D.REG"
0000    106 ;RDM_CMI                          "RDMCF6/STROBE.CMI"
0000    107 ;INH.CLOCK.SA            "RDMCF7/INH.SA.CLOCK"
0000    108
0000    109          .SBTTL  "       Diagnostic Macros"
0000    110
0000    111 ;CLOCK.EXT                        "CLKX/XTND"
0000    112 ;CLRCH.VA_RDM                     "WB_RDM,WCTRL/CLRCH.VA_WB,BUS/PRB.RD.PTE"
```

```
0000    113 ;CLRCH.VA_VA                    "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRCH.VA_WB
0000    114 ;CLRTB.VA_D                     "RSRC/ZERO,MUX/D.R1,ALU/OR,WCTRL/CLRTB.VA_WB,
0000    115 ;                                       BUS/PRB.RD.PTE"
0000    116 ;CLRTB.VA_RDM                   "WB_RDM,WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
0000    117 ;CLRTB.VA_R[]                   "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/CLRTB.VA_WB,
0000    118 ;                                       BUS/PRB.RD.PTE"
0000    119 ;CLRTB.VA_VA                    "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRTB.VA_WB
0000    120 ;CLRTB.VA_WB                    "WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
0000    121 ;D_LONLIT                       "RSRC/LONLIT,ALPCTL/WX_D_R.Q_D"
0000    122 ;D_M[]_LONLIT                   "D_LONLIT,MSRC/a1"
0000    123 ;D_VA_0                         "RSRC/ZERO,ROT/ZERO,MUX/R.S,ALU/OR,DQ1/D_WX,
0000    124 ;                                       WCTRL/VA_WB"
0000    125 ;FPA_M[] FPA_RDM                        "MSRC/a1,WB_RDM,FPA/FPA_MBUS.FPA_WBUS"
0000    126 ;FPA_M[] FPA_R[]                        "FPA/FPA_MBUS.FPA_WBUS,MSRC/a1,RSRC/a2,MUX/R
0000    127 ;                                       ROT/ZERO,ALU/OR"
0000    128 ;FPA_RDM                                "WB_RDM,FPA/FPA_DATA.WBUS"
0000    129 ;FPA_R[]                                "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,FPA/FPA_DAT
0000    130 ;FPA_R[]+M[]                    "RSRC/a1,MSRC/a2,MUX/M.R1,ALU/A+B+CI,ALUCI/ZERO,FPA/
0000    131 ;FPA_WB                         "FPA/FPA_DATA.WBUS"
0000    132 ;MEMSCAR_R[]                    "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/MEMSCAR_WB"
0000    133 ;M[]_FPA                                "MSRC/a1,SPW/MLONG,FPA/WBUS_FPA"
0000    134 ;M[]_LONLIT                     "MSRC/a1,SPW/MLONG,ALPCTL/WX_R.Q_D,RSRC/LONLIT"
0000    135 ;M[]_R[].OR.NIB0(M)             "MSRC/a1,RSRC/a2,SPW/MLONG,ROT/GETNIB,ALU/OR,
0000    136 ;                                       MUX/R.S,DQ1/NOP"
0000    137 ;M[]_RDM                                "WB_RDM,SPW/MLONG,MSRC/a1"
0000    138 ;PC_PC+RDM                      "WCTRL/PC_PC+WB,WB_RDM"
0000    139 ;PC_PC+4 MB_XB                  "MSRC/XB,PC_PC+I,ISTRM/ISIZE_DSIZE,SIZE[LONG]"
0000    140 ;PC_PC+4 RDM_XB                 "RSRC/ZERO,MSRC/XB,PC_PC+I,ISTRM/ISIZE_DSIZE,
0000    141 ;                                       MUX/M.R1,ALU/OR,RDM_WB,SIZE[LONG]"
0000    142 ;PC_RDM                         "WB_RDM,WCTRL/PC_WB"
0000    143 ;PSL_RDM                                "CCPSL/PSL_WB.CCBR_ALUS,WB_RDM"
0000    144 ;PSL_WB                         "CCPSL/PSL_WB.CCBR_ALUS"
0000    145 ;Q_D WX_R                       "ALPCTL/WX_R.Q_D"
0000    146 ;Q_LONLIT                       "RSRC/LONLIT,ALPCTL/WX_S.Q_R"
0000    147 ;Q_M[].RL.PTE                   "ALPCTL/WX_Q_S,MSRC/a1,ROT/RL.MM.PTE"
0000    148 ;Q_R[].AND.Q                    "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/AND"
0000    149 ;Q_R[].ASL.P                    "ALPCTL/WX_Q_S,RSRC/a1,ROT/ASL.R.P"
0000    150 ;Q_R[].OR.Q                     "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/OR"
0000    151 ;Q_R[].RL.P                     "ALPCTL/WX_Q_S,RSRC/a1,ROT/RL.RR.P"
0000    152 ;RDM_ALUF                       "ALPCTL/WB_ALUF,RDM_WB"
0000    153 ;RDM_FPA                                "FPA/WBUS_FPA,RDM_WB"
0000    154 ;RDM_LONLIT                     "WB_LONLIT,RDM_WB"
0000    155 ;RDM_LONLIT.Q_M                 "RSRC/LONLIT,ALPCTL/WX_R.Q_M,RDM_WB"
0000    156 ;RDM_M[]                                "MSRC/a1,ROT/ZERO,MUX/M.S,ALU/OR,RDM_WB"
0000    157 ;RDM_M[].OR.R[]                 "WB_M[a1].OR.R[a2],RDM_WB"
0000    158 ;RDM_M[]-R[]                    "WB_M[a1]-R[a2],RDM_WB"
0000    159 ;RDM_M[].XZ.OR.R[]              "ALU/A+B+CI,MUX/R.S,ROT/XZ.MM,MSRC/a1,RSRC/a2,
                                            RDM_WB"
0000    160 ;
0000    161 ;RDM_PC                         "MSRC/PC,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
0000    162 ;RDM_PCBACK                     "RSRC/ZERO,MSRC/PCBACK,MUX/M.R1,ALU/OR,RDM_WB"
0000    163 ;RDM_Q                          "RSRC/ZERO,MUX/R.Q,ALU/OR,RDM_WB"
0000    164 ;RDM_Q Q_D                      "ALPCTL/WX_Q.Q_D,RDM_WB"
0000    165 ;RDM_R[]                                "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,RDM_WB"
0000    166 ;RDM_TB                         "WB_M[TB],RDM_WB,BUS/PRB.RD,SIZE[LONG]"
0000    167 ;RDM_VA                         "MSRC/VA,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
```

```
0000   168 ;RNUM_LONLIT                    "ALPCTL/WX_D_R.Q_M,MSRC/RNUM_WBUS,RSRC/LONLIT"
0000   169 ;R[]_RDM                            "WB_RDM,SPW/RLONG,RSRC/a1"
0000   170 ;R[].DT_Q                      "RSRC/a1,D_Q Q_D,SPW/RSIZE,VSIZE/1"
0000   171 ;R[]_VA                       "RSRC/a1,MSRC/VA,ROT/ZERO,MUX/M.S,ALU/OR,SPW/RLONG"
0000   172 ;R[]_WB                       "RSRC/a1,SPW/RLONG"
0000   173 ;TB_RDM                       "WCTRL/TB_WB,WB_RDM,MSRC/VA"
0000   174 ;TB_WB                        "WCTRL/TB_WB,MSRC/VA"
0000   175 ;TESTFPA []                    "TESTFPA/a1"
0000   176 ;VA_PC+LONLIT+I.PC_PC+I       "RSRC/LONLIT,MSRC/XB.PC_PC+I,ROT/ZERO,MUX/R.S,ALU/OR
0000   177 ;                                    ISTRM/ISIZE_DSIZE,WCTRL/VA_PC+I+W.PC_PC+I,
0000   178 ;                                    SIZE[LONG]"
0000   179 ;VA_Q.OR.R[]                   "WCTRL/VA_WB,RSRC/a1,ALU/OR,MUX/R.Q"
0000   180 ;VA_RDM                       "WB_RDM,WCTRL/VA_WB"
0000   181 ;VA_R[].RL.P                  "ALPCTL/WX_S,ROT/RL.RR.P,RSRC/a1,WCTRL/VA_WB"
0000   182 ;VA_VA-ZLIT0[]                "WCTRL/VA_WB,LIT/LITRL,LITRL/a1,ROT/ZLIT0,MSRC/VA,
0000   183 ;                                    MUX/M.S,ALU/A-B-CI"
0000   184 ;VA_WB                        "WCTRL/VA_WB"
0000   185 ;WB_FPA CCR                   "FPA/WBUS_FPA.CC"
0000   186 ;WB_FPA TCR                   "WCTRL/FPTCR"
0000   187 ;WB_LONLIT                    "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
0000   188 ;WB_LONLIT.Q_M                "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
0000   189 ;WB_M[]+R[]                   "ALU/A+B+CI,ALUCI/ZERO,MUX/M.R1,MSRC/a1,RSRC/a2"
0000   190 ;WB_R[].Q_D                   "RSRC/a1,ALPCTL/WX_R.Q_D"
0000   191 ;WDR_R[]                          "WCTRL/WDR_WB,ALU/OR,MUX/R.S,RSRC/a1,ROT/ZER
0000   192 ;WRITE.LONG R[]                "BUS/WRITE.LNG,WCTRL/WDR_WB.UR,RSRC/a1,ROT/ZERO,
0000   193 ;                                    MUX/R.S,ALU/OR,SIZE[LONG]"
0000   194          .SBTTL  ""
0000   195          .SBTTL  "MODULE GATE ARRAY MAPS"
```

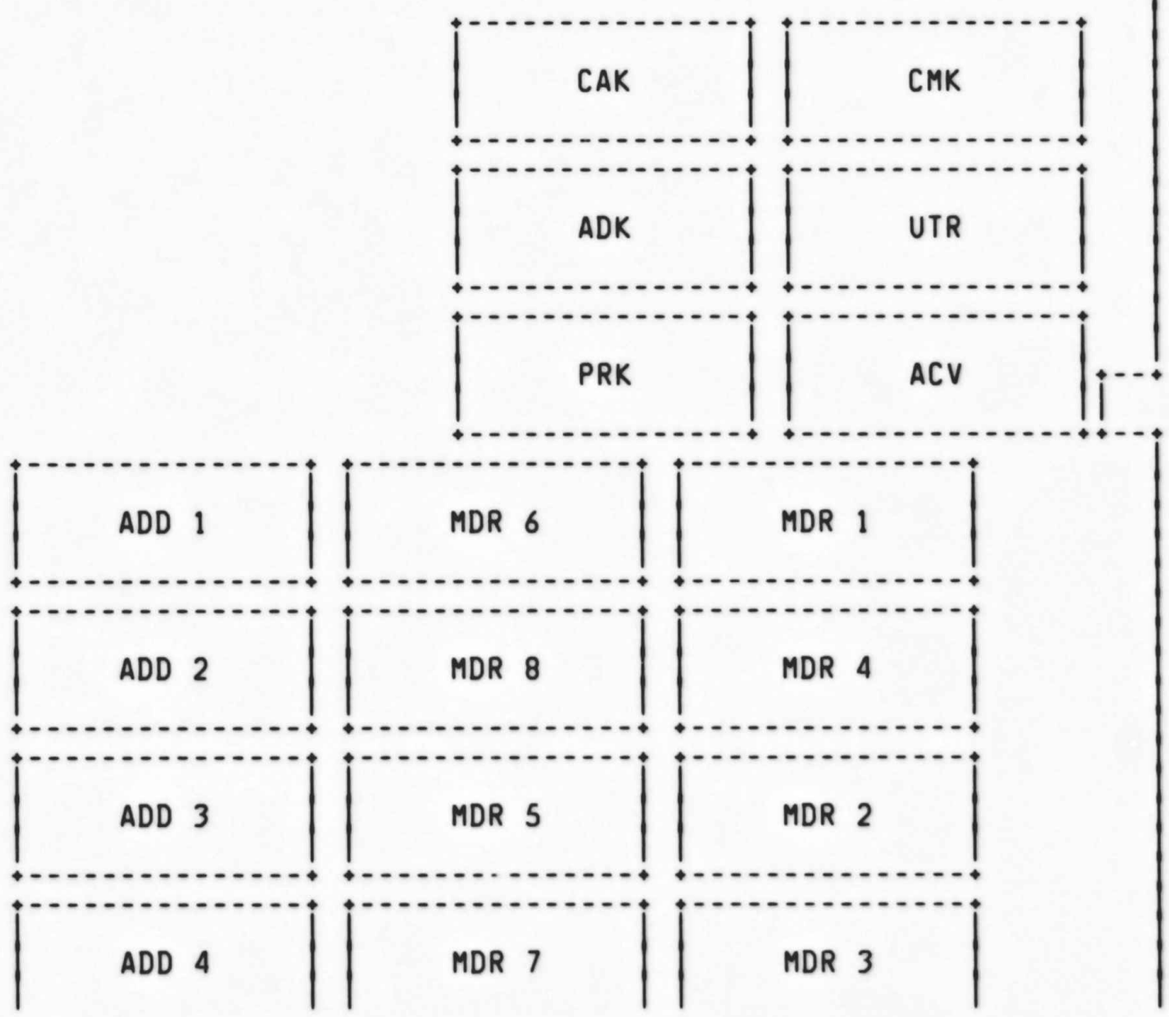
```
                        0000    197          .SBTTL  "       L0001 MULTIPLIER AND GATE ARRAY LOCATION"
                        0000    198 ;+-----------------------------------------------------------------------+
                        0000    199 ;|  +---------------+ +---------------+ +---------------+ +---------------+ |
                        0000    200 ;|  |               | |               | |               | |               | |
                        0000    201 ;|  |      FCC      | |      FQA      | |     FEX 0     | |     FEX 1     | |
                        0000    202 ;|  |               | |               | |               | |               | |
                        0000    203 ;|  +---------------+ +---------------+ +---------------+ +---------------+ |
                        0000    204 ;|                                                                       |
                        0000    205 ;|                                                                       |
                        0000    206 ;|                                                                       |
                        0000    207 ;|                                                                       |
                        0000    208 ;|                                                                       |
                        0000    209 ;|                 +---------------+ +---------------+ +---------------+ |
                        0000    210 ;|                 |               | |               | |               | |
                        0000    211 ;|                 |     FFA 5     | |     FFA 0     | |     FIO 1     | |
                        0000    212 ;|                 |               | |               | |               | |
                        0000    213 ;|                 +---------------+ +---------------+ +---------------+ |
                        0000    214 ;|  +---------------+ +---------------+ +---------------+ +---------------+ |
                        0000    215 ;|  |               | |               | |               | |               | |
                        0000    216 ;|  |     MULT 4    | |     FFA 4     | |     FCS 1     | |     FIO 0     | |
                        0000    217 ;|  |               | |               | |               | |               | |
                        0000    218 ;|  +---------------+ +---------------+ +---------------+ +---------------+ |
                        0000    219 ;|  +---------------+ +---------------+ +---------------+ +---------------+ |
                        0000    220 ;|  |               | |               | |               | |               | |
                        0000    221 ;|  |     MULT 1    | |     ALU CLA   | |     FCS 2     | |     FMR 1     | |
                        0000    222 ;|  |               | |               | |               | |               | |
                        0000    223 ;|  +---------------+ +---------------+ +---------------+ +---------------+ |
                        0000    224 ;|  +---------------+ +---------------+ +---------------+ +---------------+ |
                        0000    225 ;|  |               | |               | |               | |               | |
                        0000    226 ;|  |     MULT 5    | |     FFA 1     | |     FCS 0     | |     FMR 0     | |
                        0000    227 ;|  |               | |               | |               | |               | |
                        0000    228 ;|  +---------------+ +---------------+ +---------------+ +---------------+ |
                        0000    229 ;|  +---------------+ +---------------+ +---------------+ +---------------+ |
                        0000    230 ;|  |               | |               | |               | |               | |
                        0000    231 ;|  |     MULT 2    | |     FFA 2     | |     FCS 3     | |     FIO 2     | |
                        0000    232 ;|  |               | |               | |               | |               | |
                        0000    233 ;|  +---------------+ +---------------+ +---------------+ +---------------+ |
                        0000    234 ;|  +---------------+ +---------------+ +---------------+ +---------------+ |
                        0000    235 ;|  |               | |               | |               | |               | |
                        0000    236 ;|  |     MULT 6    | |     FFA 6     | |     FIO 3     | |     FIO 5     | |
                        0000    237 ;|  |               | |               | |               | |               | |
                        0000    238 ;|  +---------------+ +---------------+ +---------------+ +---------------+ |
                        0000    239 ;|  +---------------+ +---------------+ +---------------+ +---------------+ |
                        0000    240 ;|  |               | |               | |               | |               | |
                        0000    241 ;|  |     MULT 3    | |     FFA 3     | |     INC CLA   | |     FIO 4     | |
                        0000    242 ;|  |               | |               | |               | |               | |
                        0000    243 ;|  +---------------+ +---------------+ +---------------+ +---------------+ |
                        0000    244 ;|  +---------------+ +---------------+ +---------------+ +---------------+ |
                        0000    245 ;|  |               | |               | |               | |               | |
                        0000    246 ;|  |     MULT 7    | |     FFA 7     | |     FIO 7     | |     FIO 6     | |
                        0000    247 ;|  |               | |               | |               | |               | |
                        0000    248 ;|  +---------------+ +---------------+ +---------------+ +---------------+ |
                        0000    249 ;|                                                                       |
                        0000    250 ;|                                                                       |
                        0000    251 ;|                                                                       |
```

```
                0000   252 ;+-----------------------------------------------------------------------------+
```

```
                        0000    254          .SBTTL  "       L0002 GATE ARRAY LOCATION"
                        0000    255 ;+-----------------------------------------------------------------------+
                        0000    256 ;|                                                                       |
                        0000    257 ;|                                                                       |
                        0000    258 ;|                                                                       |
                        0000    259 ;|                                                                       |
                        0000    260 ;|                                                                       |
                        0000    261 ;|                                                                       |
                        0000    262 ;|                                                                       |
                        0000    263 ;|                                                                       |
                        0000    264 ;|                 +---------------+ +---------------+                   |
                        0000    265 ;|                 |               | |               |                   |
                        0000    266 ;|                 |      PHB      | |      MSQ      |                   |
                        0000    267 ;|                 |               | |               |                   |
                        0000    268 ;|                 +---------------+ +---------------+                   |
                        0000    269 ;|                 +---------------+ +---------------+                   |
                        0000    270 ;|                 |               | |               |                   |
                        0000    271 ;|                 |      CCC      | |      SAC      |                   |
                        0000    272 ;|                 |               | |               |                   |
                        0000    273 ;|                 +---------------+ +---------------+                   |
                        0000    274 ;|  +---------------+ +---------------+ +---------------+                 |
                        0000    275 ;|  |               | |               | |               |                 |
                        0000    276 ;|  |      SPA      | |      SRK      | |      IRD      |                 |
                        0000    277 ;|  |               | |               | |               |                 |
                        0000    278 ;|  +---------------+ +---------------+ +---------------+                 |
                        0000    279 ;|  +---------------+ +---------------+ +---------------+                 |
                        0000    280 ;|  |               | |               | |               |                 |
                        0000    281 ;|  |      TOK      | |      CLA      | |      ALK      |                 |
                        0000    282 ;|  |               | |               | |               |                 |
                        0000    283 ;|  +---------------+ +---------------+ +---------------+                 |
                        0000    284 ;|                 +---------------+ +---------------+                   |
                        0000    285 ;|                 |               | |               |                   |
                        0000    286 ;|                 |     ALP 1     | |     ALP 5     |                   |
                        0000    287 ;|                 |               | |               |                   |
                        0000    288 ;|                 +---------------+ +---------------+                   |
                        0000    289 ;|                 +---------------+ +---------------+                   |
                        0000    290 ;|                 |               | |               |                   |
                        0000    291 ;|                 |     ALP 2     | |     ALP 6     |                   |
                        0000    292 ;|                 |               | |               |                   |
                        0000    293 ;|                 +---------------+ +---------------+                   |
                        0000    294 ;|                 +---------------+ +---------------+                   |
                        0000    295 ;|                 |               | |               |                   |
                        0000    296 ;|                 |     ALP 3     | |     ALP 7     |                   |
                        0000    297 ;|                 |               | |               |                   |
                        0000    298 ;|                 +---------------+ +---------------+                   |
                        0000    299 ;|                 +---------------+ +---------------+                   |
                        0000    300 ;|                 |               | |               |                   |
                        0000    301 ;|                 |     ALP 4     | |     ALP 8     |                   |
                        0000    302 ;|                 |               | |               |                   |
                        0000    303 ;|                 +---------------+ +---------------+                   |
                        0000    304 ;|  +---------------+ +---------------+ +---------------+ +---------------+ |
                        0000    305 ;|  |               | |               | |               | |               | |
                        0000    306 ;|  |     SRM 1     | |     SRM 2     | |     SRM 3     | |     SRM 4     | |
                        0000    307 ;|  |               | |               | |               | |               | |
                        0000    308 ;|  +---------------+ +---------------+ +---------------+ +---------------+ |
```

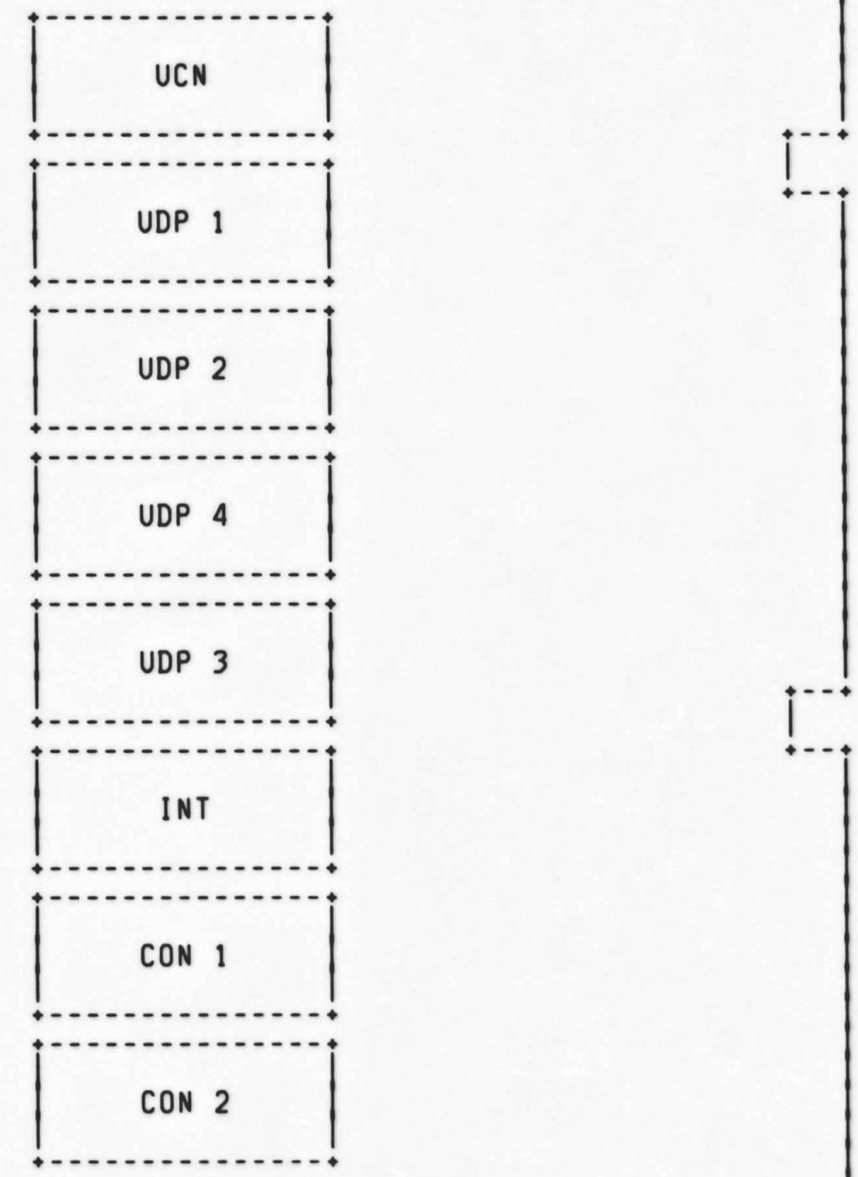
```
                        0000   309 ;+-----------------------------------------------------------------------------+
```

```
                                0000    311          .SBTTL  "       L0003 GATE ARRAY LOCATION"
                                0000    312 ;+-----------------------------------------------------------------------------+
                                0000    313 ;|                                                                             |
                                0000    314 ;|                                                                             |
                                0000    315 ;|                                                                             |
                                0000    316 ;|                                                                             |
                                0000    317 ;|                                                                             |
                                0000    318 ;|                                                                             |
                                0000    319 ;|                                                                             |
                                0000    320 ;|                                                                             |
                                0000    321 ;|                                                                             |
                                0000    322 ;|                                                                             |
                                0000    323 ;|                                                                             |
                                0000    324 ;|                                                                             |
                                0000    325 ;|                                                                             |
                                0000    326 ;|                                                                             |
                                0000    327 ;|                                                                             |
                                0000    328 ;|                                                                             |
                                0000    329 ;|                                                                             |
                                0000    330 ;|                                                                             |
                                0000    331 ;|                                                                             |
                                0000    332 ;|                                                                             |
                                0000    333 ;|                                                                             |
                                0000    334 ;|                                                                             |
                                0000    335 ;|                                                                             |
                                0000    336 ;|                                                                             |
                                0000    337 ;|                                                                             |
                                0000    338 ;|                                                                             |
                                0000    339 ;|                                                                             |
                                0000    340 ;|                                                                             |
                                0000    341 ;|                                                                             |
                                0000    342 ;|                                                                             |
                                0000    343 ;|                                                                             |
                                0000    344 ;|                                                                             |
                                0000    345 ;|                                                                             |
                                0000    346 ;|                                                                             |
                                0000    347 ;|                                                                             |
                                0000    348 ;|                                                                             |
                                0000    349 ;|                                                                             |
                                0000    350 ;|                                                                             |
                                0000    351 ;|                                                                             |
                                0000    352 ;|                                                                             |
                                0000    353 ;|                                                                             |
                                0000    354 ;|                                                                             |
                                0000    355 ;|                                                                             |
                                0000    356 ;|                                                                             |
                                0000    357 ;|                                                                             |
                                0000    358 ;|                                                                             |
                                0000    359 ;|                                                                             |
                                0000    360 ;|                                                                             |
                                0000    361 ;|                                                                             |
                                0000    362 ;|                                                                             |
                                0000    363 ;|                                                                             |
                                0000    364 ;|                                                                             |
                                0000    365 ;|                                                                             |
```



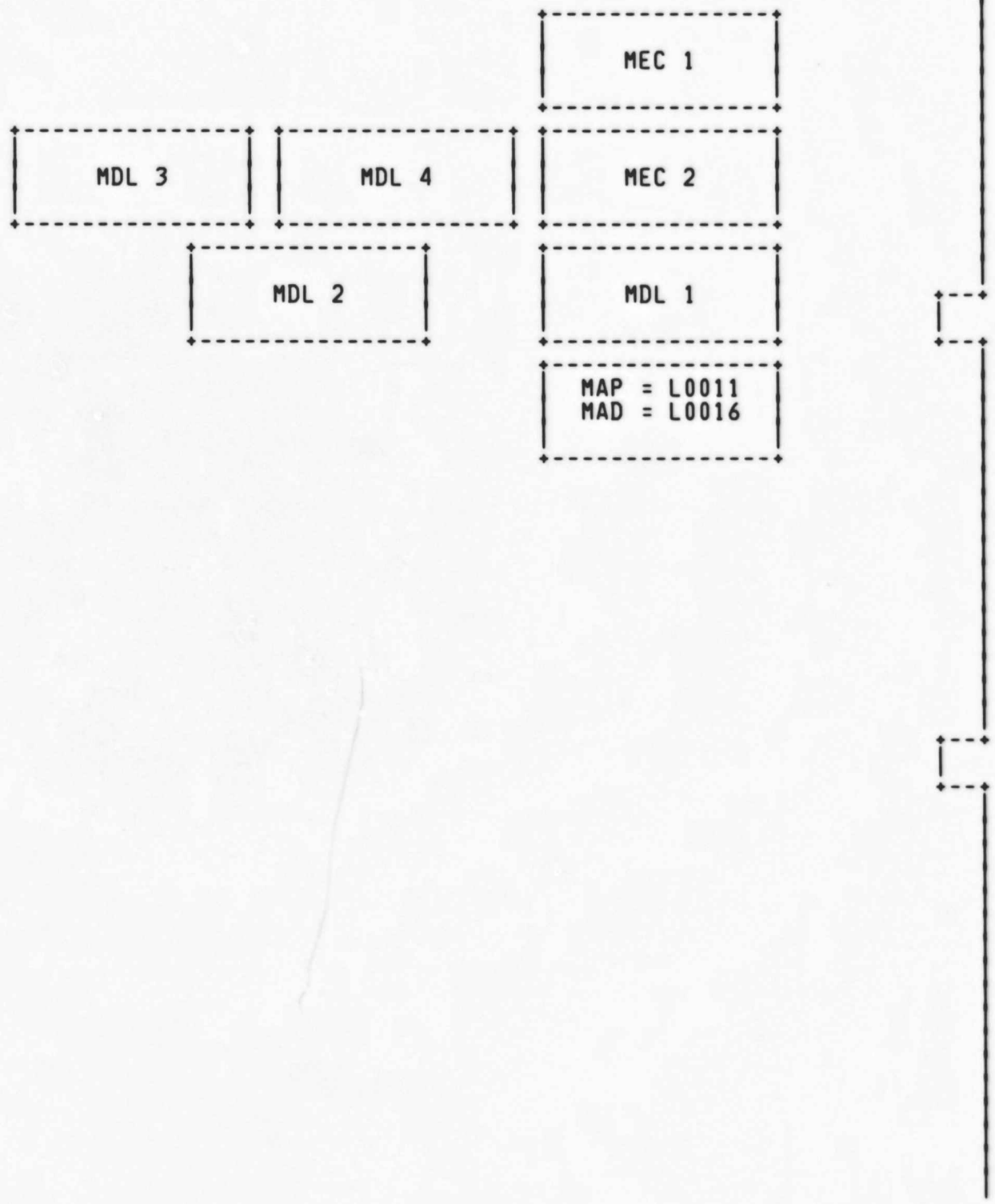
```
0000   366 ;|              +---------------+ +---------------+ +---------------+          |
0000   367 ;+-------------------------------------------------------------------------------+
```

```
                        .SBTTL  "       L0004 GATE ARRAY LOCATION"
```



```
0000   424 ;|                                                                                |
0000   425 ;+--------------------------------------------------------------------------------+
```

```
                        427          .SBTTL  "       L0011 & L0016 GATE ARRAY LOCATION"
                        428 ;+-----------------------------------------------------------------------------+
                        429 ;|                                                                             |
                        430 ;|                                                                             |
                        431 ;|                                                                             |
                        432 ;|                                                                             |
                        433 ;|                                                                             |
                        434 ;|                                                                             |
                        435 ;|                                                                             |
                        436 ;|                                                                             |
                        437 ;|                                                                             |
                        438 ;|                                                                             |
                        439 ;|                                                                             |
                        440 ;|                                                                             |
                        441 ;|                                                                             |
                        442 ;|                                                                             |
                        443 ;|                                                                             |
                        444 ;|                                                                             |
                        445 ;|                                                                             |
                        446 ;|                                                                             |
                        447 ;|                                                                             |
                        448 ;|                                                                             |
                        449 ;|                                                                             |
                        450 ;|                                                                             |
                        451 ;|                                                                             |
                        452 ;|                                                                             |
                        453 ;|                                                                             |
                        454 ;|                                                                             |
                        455 ;|                                                                             |
                        456 ;|                                                                             |
                        457 ;|                                                                             |
                        458 ;|                                                                             |
                        459 ;|                                                                             |
                        460 ;|                                                                             |
                        461 ;|                                                                             |
                        462 ;|                                                                             |
                        463 ;|                                                                             |
                        464 ;|                                                                             |
                        465 ;|                                                                             |
                        466 ;|                                                                             |
                        467 ;|                                                                             |
                        468 ;|                                                                             |
                        469 ;|                                                                             |
                        470 ;|                                                                             |
                        471 ;|                                                                             |
                        472 ;|                                                                             |
                        473 ;|                                                                             |
                        474 ;|                                                                             |
                        475 ;|                                                                             |
                        476 ;|                                                                             |
                        477 ;|                                                                             |
                        478 ;|                                                                             |
                        479 ;|                                                                             |
                        480 ;|                                                                             |
                        481 ;|                                                                             |
```



```
0000   482 ;|                                                                    |
0000   483 ;+-------------------------------------------------------------------+
```

```
                        0000   485          .SBTTL  ""
                        0000   486          .LIST   MC
                        0000   487          .LIBRARY        /UCOD$0:[VAX750]COMETMAC/
                        0000   488          .LIST   ME
                        0000   489          .NLIST  CND,MC
                        0000                .SBTTL  "EQUATED SYMBOLS
                        0000         ;
                        0000         ; RDM REGISTER DEFINITIONS:
                        0000         ;
                        0000         ; NOTE: THESE REGITER DEFINITIONS ARE OFFSET BY THE LOAD/START ADDRESS OF
                        0000         ;       THE PROGRAM. FOR THE SIMULATOR VERSION THIS IS 400(X) AND FOR THE
                        0000         ;       REAL VERSION IT IS 8400(X).
                        0000         ;
00007400                0000                DCS_DATA_REG    =       ^XF800 + HEAD
00007401                0000                DCS_ADDR_REG_WO =       ^XF801 + HEAD
00007402                0000                DCS_CTRL_REG    =       ^XF802 + HEAD
00007403                0000                DCS_CTRL_FILE   =       ^XF803 + HEAD
00007404                0000                ADDR_MATCH_LO   =       ^XF804 + HEAD
00007405                0000                ADDR_MATCH_HI   =       ^XF805 + HEAD
00007406                0000                STATUS_REG      =       ^XF806 + HEAD
                        0000         ;
00007410                0000                A_REG_BYTE_0    =       ^XF810 + HEAD
00007411                0000                A_REG_BYTE_1    =       ^XF811 + HEAD
00007412                0000                A_REG_BYTE_2    =       ^XF812 + HEAD
00007413                0000                A_REG_BYTE_3    =       ^XF813 + HEAD
00007414                0000                MD_REG_BYTE_0   =       ^XF814 + HEAD
00007415                0000                MD_REG_BYTE_1   =       ^XF815 + HEAD
00007416                0000                MD_REG_BYTE_2   =       ^XF816 + HEAD
00007417                0000                MD_REG_BYTE_3   =       ^XF817 + HEAD
00007418                0000                MH_REG_BYTE_0   =       ^XF818 + HEAD
00007419                0000                MH_REG_BYTE_1   =       ^XF819 + HEAD
0000741A                0000                MH_REG_BYTE_2   =       ^XF81A + HEAD
0000741B                0000                MH_REG_BYTE_3   =       ^XF81B + HEAD
0000741C                0000                CMI_CTRL_REG    =       ^XF81C + HEAD
0000741D                0000                MAINT_REG       =       ^XF81D + HEAD
0000741E                0000                RD_CTRL_REG     =       ^XF81E + HEAD
                        0000         ;
00007420                0000                FRONT_PNL_1     =       ^XF820 + HEAD
00007421                0000                FRONT_PNL_2     =       ^XF821 + HEAD
00007422                0000                CS_ADD_TRAP_LOW =       ^XF822 + HEAD
00007423                0000                CS_ADD_TRAP_HGH =       ^XF823 + HEAD
00007424                0000                CS_ADD_BUFF_LOW =       ^XF824 + HEAD
00007425                0000                CS_ADD_BUFF_HGH =       ^XF825 + HEAD
00007426                0000                BUFF_ADDR_LOW   =       ^XF826 + HEAD
00007427                0000                DCS_ADDR_REG_RO =       ^XF827 + HEAD
                        0000         ;
0000742C                0000                WD_REG_BYTE_0   =       ^XF82C + HEAD
0000742D                0000                WD_REG_BYTE_1   =       ^XF82D + HEAD
0000742E                0000                WD_REG_BYTE_2   =       ^XF82E + HEAD
0000742F                0000                WD_REG_BYTE_3   =       ^XF82F + HEAD
00007430                0000                WH_REG_BYTE_0   =       ^XF830 + HEAD
00007431                0000                WH_REG_BYTE_1   =       ^XF831 + HEAD
00007432                0000                WH_REG_BYTE_2   =       ^XF832 + HEAD
00007433                0000                WH_REG_BYTE_3   =       ^XF833 + HEAD
                        0000         ;
```

```
                        0000        ; RDM REGISTER NAMES AS DEFINED IN HARDWARE SPEC
                        0000        ;
                00007400 0000               DCSDA=DCS_DATA_REG              ;Diagnostic control store data register
                00007401 0000               DCSAD=DCS_ADDR_REG_WO           ;Diagnostic control store address reg
                00007402 0000               DCSCR=DCS_CTRL_REG              ;Diagnostic control store control reg
                00007403 0000               DCSCF=DCS_CTRL_FILE             ;Diagnostic control store control file
                00007404 0000               CSAMR=ADDR_MATCH_LO             ;Control store address match register
                00007406 0000               STATUS=STATUS_REG               ;Status register
                00007410 0000               MAREG=A_REG_BYTE_0              ;Memory address register
                00007414 0000               MDREG=MD_REG_BYTE_0             ;Memory data register
                00007418 0000               MHREG=MH_REG_BYTE_0             ;Memory holding register
                0000741C 0000               CMICTL=CMI_CTRL_REG             ;CMI control
                0000741D 0000               MAIN32=MAINT_REG                ;32 Bit path maintenance control
                0000741E 0000               RDCTRL=RD_CTRL_REG              ;Remote diagnosis control
                00007420 0000               FRONT1=FRONT_PNL_1              ;Front panel readback 1
                00007421 0000               FRONT2=FRONT_PNL_2              ;Front panel readback 2
                00007422 0000               CSTRP=CS_ADD_TRAP_LOW           ;Control store address trap register
                00007424 0000               CSBUF=CS_ADD_BUFF_LOW           ;Control store address buffer
                00007426 0000               BADD1=BUFF_ADDR_LOW             ;Control store address trace readback
                00007427 0000               CLKDCS=DCS_ADDR_REG_RO          ;Clock control & DCS address register
                0000742C 0000               WDREG=WD_REG_BYTE_0             ;WBUS data register
                00007430 0000               WHREG=WH_REG_BYTE_0             ;WBUS holding register
                        0000        ;
                        0000        ; RDM REGISTER BIT DEFINITIONS:
                        0000        ;
                        0000        ; DCS CONTROL REGISTER
                00000001 0000               HALT_ON_MATCH    =      1
                00000002 0000               CLEAR_CTRL_FILE  =      2
                00000004 0000               TRACE_ENABL      =      4
                00000008 0000               PAR_CHK_ENABL    =      8
                00000010 0000               PAR_STOP_ENABL   =      ^X10
                00000020 0000               CLK_CTRL_0       =      ^X20    ; ACTIVE LOW
                00000040 0000               CLK_CTRL_1       =      ^X40    ; ACTIVE LOW
                00000080 0000               MICRO_ADDR_INH   =      ^X80
                        0000            ;
                        0000            ; FOLLOWING IS A FUNCTION TABLE OF THE CLOCK CONTROL BITS
                        0000            ;
                        0000                ;       CLK_CTRL 1,0    FUNCTION
                        0000                ;           11            HALT
                        0000                ;           01            SINGLE MICRO INSTRUCTION
                        0000                ;           10            SINGLE TICK
                        0000                ;           00            RUN
                        0000        ;
                        0000        ; DCS CONTROL FILE      ALL OF THESE BITS ARE ACTIVE LOW
                        0000        ;
                00000001 0000               V_BUS_STROBE     =      1
                00000002 0000               DCS_ADDR_CLEAR   =      2
                00000004 0000               STOP_CLOCK       =      4
                00000008 0000               EN_TRACE_REG     =      8
                00000010 0000               H_FROM_WBUS      =      ^X10
                00000020 0000               WBUS_FROM_D      =      ^X20
                00000040 0000               STROBE_CMI       =      ^X40
                00000080 0000               SA_CLOCK         =      ^X80
                        0000        ;
                        0000        ; CS ADDRESS MATCH REGISTER (HIGH)
```

```
                    0000        ;
00000040            0000                VBUS_SERIAL_IN  =       ^X40    ; ACTIVE LOW
00000080            0000                VBUS_CLOCK      =       ^X80
                    0000        ;
                    0000        ; STATUS REGISTER
                    0000        ;
00000001            0000                VBUS_SERIAL_OUT =       1
00000002            0000                TRAP_OFF        =       2
00000008            0000                CMI_STATUS_0    =       8
00000010            0000                CMI_STATUS_1    =       ^X10
                    0000                ;
                    0000                ; THE CMI STATUS BITS ARE ENCODED AS FOLLOWS:
                    0000                ;
                    0000                ;       00 = NO RESPONSE
                    0000                ;       01 = UNCORRECTABLE ERROR
                    0000                ;       10 = CORRECTABLE ERROR
                    0000                ;       11 = NO ERRORS
00000020            0000                CMI_COMPLETE    =       ^X20
                    0000        ;
00000080            0000                CLOCK_STOPED    =       ^X80
                    0000        ;
                    0000        ; CMI CONTROL REGISTER
                    0000        ;
00000001            0000                CMI_CTRL_CLEAR  =       1
00000008            0000                CMI_GO          =       8
00000020            0000                CMI_WRITE       =       ^X20
00000040            0000                CMI_READ        =       ^X40
00000080            0000                SA_START_STOP   =       ^X80
                    0000        ;
                    0000        ; 32 BIT MAINTENANCE CONTROL REGISTER
                    0000        ;
00000001            0000                MAINT_TRAP_OFF  =       1
00000002            0000                MAINT_STB_INH   =       2
00000004            0000                MAINT_DCS_ENABL =       4
00000008            0000                MAINT_WB_TO_WH  =       8
00000010            0000                MAINT_WD_TO_WB  =       ^X10
00000020            0000                MAINT_CMI_TO_MH =       ^X20
00000040            0000                MAINT_MD_TO_CMI =       ^X40
00000080            0000                MAINT_A_TO_CMI  =       ^X80
                    0000        ;
                    0000        ; RD CONTROL REGISTER
                    0000        ;
00000008            0000                MASTER_HALT_EN  =       8
00000010            0000                VBUS_LOAD       =       ^X10
00000020            0000                TRAP_HALT_EN    =       ^X20
00000040            0000                DC_LOW          =       ^X40
00000080            0000                AC_LOW          =       ^X80
                    0000        ;
                    0000        ; FRONT PANEL REGISTER 1
                    0000        ;
00000001            0000                FP_BOOT_0       =       1
00000002            0000                FP_BOOT_1       =       2
00000004            0000                FP_START_0      =       4
00000008            0000                FP_START_1      =       8
00000010            0000                CPU_RUN         =       ^X10
```

```
00000020  0000                PARITY_ERROR    =       ^X20
          0000        ;
          0000        ; FRONT PANEL REGISTER 2
          0000        ;
00000001  0000                REMOTE          =       1
00000002  0000                FAULT           =       2
00000004  0000                TEST            =       4
          0000        ;
          0000        ;
          0000        ;
00000040  0000                PB_INIT         =       ^X40    ; ACTIVE LOW
00000080  0000                FRONT_PNL_LOCK  =       ^X80
          0000        ;
          0000        ; DCS ADDRESS REGISTER READ ONLY
          0000        ;
00000040  0000                CLK_CTRL_0_RO   =       ^X40    ; ACTIVE HIGH
00000080  0000                CLK_CTRL_1_RO   =       ^X80    ; ACTIVE HIGH
          0000        ;
          0000        ; MISCELLANEOUS EQUATES:
          0000        ;
00000001  0000                BYTE            =       1
00000002  0000                WORD            =       2
00000004  0000                LONG            =       4
0000000A  0000                MICROWORD       =       10
00002C5E  0000                MONITOR_SIZE    =       ^X2C5E
00000400  0000                STACK_SIZE      =       1024
000007A2  0000                M$TEST_SIZE     =       14336 - STACK_SIZE - MONITOR_SIZE
          0000                                              ; MAXIMUM SIZE OF A TEST OVERLAY
000003E2  0000                M$TEST_SIZE_D   = M$TEST_SIZE - 960
          0000                                              ; MAXIMUM SIZE OF TEST OVERLAY WITH
          0000                                              ; 'DEBUG' ENABLED IN
00000001  0000                B               =       BYTE
00000002  0000                W               =       WORD
00000004  0000                L               =       LONG
0000000A  0000                M               =       MICROWORD
00000001  0000                HIGH            =       1
00000000  0000                LOW             =       0
00000000  0000                ONERROR         =       0
00000001  0000                NOERROR         =       1
00000002  0000                ONEQUAL         =       2
00000003  0000                NOTEQUAL        =       3
00000010  0000                MAX_ARGUMENTS   =       16      ; BYTE LENGTH OF LARGEST PSEUDO OP
00000036  0000                DCS_SCRATCH_ADR =       54      ; DCS SCRATCH ADDRESS START
000032FC  0000                ERROR_LOG_ADR   =       ^XB6FC + HEAD
          0000                                              ; START ADDRESS OF MEMORY ERROR LOG
          0000    491   .NLIST  ME
          0000    492   .LIST   MC
          0000    493   .SBTTL  ""
          0000    494   .NLIST  MC      ; SHUT-OFF LISTING FOR FIRST 'NEWTEST' CALL
```

```
0045          .SBTTL  "TEST   037     Rotator Test  5 PLATCH writes to SBUS when SLATCH gets WBUS
0045       3 ;*******************************************************************************
0045       4 ;++
0045       5 ;
0045       6 ; FUNCTIONAL DESCRIPTION
0045       7 ;
0045       8 ;       This test checks to see if the PLATCH writes to the SBUS when the
0045       9 ;       SLATCH is written from the WBUS.
0045      10 ;
0045      11 ; TEST ALGORITHM
0045      12 ;
0045      13 ;       1. Setup a loop of six iterations.
0045      14 ;       2. Generate a standard stuck at zero or one test pattern.
0045      15 ;       3. Load the pattern into the RDM's WDREG.
0045      16 ;       4. Execute the micro-sequence:
0045      17 ;               1. Zero the PLATCH.
0045      18 ;               2. Load the PLATCH with the test pattern.
0045      19 ;               3. Use the ALPCTL field to enable the SBUS onto the WBUS while
0045      20 ;                  the rotator is performing a PL.SL_WB micro-order.
0045      21 ;               4. Read the SLATCH into the RDM.
0045      22 ;       5. Check the result.
0045      23 ;       6. Loop back until done.
0045      24 ;
0045      25 ; TEST DATA
0045      26 ;
0045      27 ;       This test uses the lower six bits of the standard stuck at zero or one
0045      28 ;       test patterns.
0045      29 ;
0045      30 ;           Pattern Generated           Expected SLATCH contents
0045      31 ;
0045      32 ;               AAAAAAAA                        2A
0045      33 ;               55555555                        15
0045      34 ;               33333333                        33
0045      35 ;               0F0F0F0F                        0F
0045      36 ;               00FF00FF                        3F
0045      37 ;               0000FFFF                        3F
0045      38 ;
0045      39 ; LOGICAL DESCRIPTION
0045      40 ;
0045      41 ;       The logic used in this test is inside the SRK gate array.
0045      42 ;
0045      43 ; ASSUMPTIONS
0045      44 ;
0045      45 ;       This test requires an operational RBUS, SBUS and WBUS.
0045      46 ;       The ALPCTL special function WMUX <- BMUX must also work.
0045      47 ;
0045      48 ; ERROR DESCRIPTION
0045      49 ;
0045      50 ;       On error this test will call out the SRK gate array.
0045      51 ;       The following information will also be typed out:
0045      52 ;
0045      53 ;       EXPECTED SLATCH DATA
0045      54 ;       RECEIVED SLATCH DATA
0045      55 ;       LOOP COUNT
0045      56 ;
```

```
                        0045    57 ;--
                        0045    58 ;*******************************************************************************
                        0045    59 ;///////////////////////////////////////////////////////////////////////////////
                        0045    60
                        0045    61         INITIALIZE                                  ; INTIALIZE THE CPU
                        0047    62         LOOP            I,1,6                       ; LOOP FOR SIX SA01 PATTERNS
                        004E    63         SA01PAT         I,1$                        ; GENERATE A SA01 TEST PATTERN
                        0056    64         LOADREG         WDREG,1$,,L                 ; LOAD THE PATTERN INTO WDREG
                        005E    65         ERRLOOP                                     ; LOOP ON ERROR FORM HERE
                        0060    66         FETCH           0                           ; WRITE THE PATTERN FROM PLATCH
                        0065    67         BRSTCLK         4                           ; TO SLATCH AND THEN INTO RDM
                        0069    68         CMPREGMSK       WHREG,2$,I,3$,,B            ; CHECK THE RESULT
                        0075    69         IFERROR         ,<SRK>                      ; ERROR CALLOUT
                        007B    70         ENDLOOP         I                           ; CONTINUE LOOP
                        007E    71         SKIP                                        ; END OF TEST
                        0085    72
                        0085    73 BEGIN_TEST_DATA
                        0085
                        0085        ;-------------------------------------------------------------------
                        0085    74
               00000000 0085    75 1$:      .LONG   0       ; LOCATION TO HOLD SA01 PATTERN
                        0089    76
                     2A 0089    77 2$:      .BYTE   ^X2A    ; EXPECTED SLATCH DATA
                     15 008A    78          .BYTE   ^X15
                     33 008B    79          .BYTE   ^X33
                     0F 008C    80          .BYTE   ^X0F
                     3F 008D    81          .BYTE   ^X3F
                     3F 008E    82          .BYTE   ^X3F
                        008F    83
                     3F 008F    84 3$:      .BYTE   ^X3F    ; MASK FOR DATA COMPARISON
                        0090    85
                        0090    86 BEGIN_CPU_DATA
                        0002    87
                        0002    88 DCS_DATA
                        0001    89
U 00, 7700,C800,0364,0300,0470,1801 0001    90 ;% 00   NOP                             ; GET CPU UP TO SPEED
U 01, 5700,8800,BF64,0300,0470,1802 000C    94 ;% 01   WB_RDM,ROT/SL.PL_WB             ; PLATCH = WDREG
U 02, 7700,0800,B747,0300,0470,1803 0017    98 ;% 02   ROT/PL.SL_WB,ALPCTL/WX_S.Q_0    ; SLATCH = PLATCH
U 03, 6700,8800,BB47,0300,0470,1804 0022   102 ;% 03   ROT/SL,ALPCTL/WX_S.Q_0,RDM_WB    ; READ SLATCH INTO RDM
U 04, 7300,4800,0364,0300,0470,1805 002D   106 ;% 04   STOP.CLOCK                       ; HALT
                        0038   110
                        0038   111 END_CPU_DATA
                        0090   112
                        0090   113 END_TEST_DATA
                        0090
                        0090        ;-------------------------------------------------------------------
                        0090   114
                        0090   115 ENDTEST
```

```
0045          .SBTTL  "TEST   038     Rotator Test  6 SLATCH writes to SBUS when PLATCH gets WBUS
0045    117 ;*****************************************************************************
0045    118 ;++
0045    119 ;
0045    120 ; FUNCTIONAL DESCRIPTION
0045    121 ;
0045    122 ;       This test checks to see if the SLATCH writes to the SBUS when the
0045    123 ;       PLATCH is written from the WBUS.
0045    124 ;
0045    125 ; TEST ALGORITHM
0045    126 ;
0045    127 ;       1. Setup a loop of six iterations.
0045    128 ;       2. Generate a standard stuck at zero or one test pattern.
0045    129 ;       3. Load the pattern into the RDM's WDREG.
0045    130 ;       4. Execute the micro-sequence:
0045    131 ;               1. Zero the SLATCH.
0045    132 ;               2. Load the SLATCH with the test pattern.
0045    133 ;               3. Use the ALPCTL field to enable the SBUS onto the WBUS while
0045    134 ;                  the rotator is performing a SL.PL_WB micro-order.
0045    135 ;               4. Read the PLATCH into the RDM.
0045    136 ;       5. Check the result.
0045    137 ;       6. Loop back until done.
0045    138 ;
0045    139 ; TEST DATA
0045    140 ;
0045    141 ;       This test uses the lower six bits of the standard stuck at zero or one
0045    142 ;       test patterns.
0045    143 ;
0045    144 ;           Pattern Generated           Expected PLATCH contents
0045    145 ;
0045    146 ;               AAAAAAAA                        2A
0045    147 ;               55555555                        15
0045    148 ;               33333333                        33
0045    149 ;               0F0F0F0F                        0F
0045    150 ;               00FF00FF                        3F
0045    151 ;               0000FFFF                        3F
0045    152 ;
0045    153 ; LOGICAL DESCRIPTION
0045    154 ;
0045    155 ;       The logic used in this test is inside the SRK gate array.
0045    156 ;
0045    157 ; ASSUMPTIONS
0045    158 ;
0045    159 ;       This test requires an operational RBUS, SBUS and WBUS.
0045    160 ;       The ALPCTL special function WMUX <- BMUX must also work.
0045    161 ;
0045    162 ; ERROR DESCRIPTION
0045    163 ;
0045    164 ;       On error this test will call out the SRK gate array.
0045    165 ;       The following information will also be typed out:
0045    166 ;
0045    167 ;       EXPECTED PLATCH DATA
0045    168 ;       RECEIVED PLATCH DATA
0045    169 ;       LOOP COUNT
0045    170 ;
```

```
                                  0045   ;--
                                  0045   ;*****************************************************************************
                                  0045   ;/////////////////////////////////////////////////////////////////////////////
                                  0045
                                  0045           INITIALIZE                                  ; INTIALIZE THE CPU
                                  0047           LOOP            I,1,6                       ; LOOP FOR SIX SA01 PATTERNS
                                  004E           SA01PAT         I,1$                        ; GENERATE A SA01 TEST PATTERN
                                  0056           LOADREG         WDREG,1$,,L                 ; LOAD THE PATTERN INTO WDREG
                                  005E           ERRLOOP                                     ; LOOP ON ERROR FORM HERE
                                  0060           FETCH           0                           ; WRITE THE PATTERN FROM SLATCH
                                  0065           BRSTCLK         4                           ; TO PLATCH AND THEN INTO RDM
                                  0069           CMPREGMSK       WHREG,2$,I,3$,,B            ; CHECK THE RESULT
                                  0075           IFERROR         ,<SRK>                      ; ERROR CALLOUT
                                  007B           ENDLOOP         I                           ; CONTINUE LOOP
                                  007E           SKIP                                        ; END OF TEST
                                  0085
                                  0085   BEGIN_TEST_DATA
                                  0085
                                  0085       ;-----------------------------------------------------------------------
                                  0085
                        00000000  0085   1$:      .LONG   0       ; LOCATION TO HOLD SA01 PATTERN
                                  0089
                              2A  0089   2$:      .BYTE   ^X2A    ; EXPECTED PLATCH DATA
                              15  008A           .BYTE   ^X15
                              33  008B           .BYTE   ^X33
                              0F  008C           .BYTE   ^X0F
                              3F  008D           .BYTE   ^X3F
                              3F  008E           .BYTE   ^X3F
                                  008F
                              3F  008F   3$:      .BYTE   ^X3F    ; MASK FOR DATA COMPARISON
                                  0090
                                  0090   BEGIN_CPU_DATA
                                  0002
                                  0002   DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801  0001   ;% 00   NOP                                       ; GET CPU UP TO SPEED
U 01, 5700,C800,B764,0300,0470,1802  000C   ;% 01   WB_RDM,ROT/PL.SL_WB                       ; SLATCH = WDREG
U 02, 7700,4800,BF47,0300,0470,1803  0017   ;% 02   ROT/SL.PL_WB,ALPCTL/WX_S.Q_0              ; PLATCH = SLATCH
U 03, 6700,C800,B347,0300,0470,1804  0022   ;% 03   ROT/PL,ALPCTL/WX_S.Q_0,RDM_WB             ; WHREG = PLATCH
U 04, 7300,4800,0364,0300,0470,1805  002D   ;% 04   STOP.CLOCK                                ; HALT
                                  0038
                                  0038   END_CPU_DATA
                                  0090
                                  0090   END_TEST_DATA
                                  0090
                                  0090       ;-----------------------------------------------------------------------
                                  0090
                                  0090   ENDTEST
```

```
                        002D        .SBTTL  "TEST   039     Rotator Test  7 PLATCH gets Literal Test
                        002D    231 ;*****************************************************************************
                        002D    232 ;++
                        002D    233 ;
                        002D    234 ; FUNCTIONAL DESCRIPTION
                        002D    235 ;
                        002D    236 ;       This test checks the data path from the LITRL field to the PLATCH.
                        002D    237 ;
                        002D    238 ; TEST ALGORITHM
                        002D    239 ;
                        002D    240 ;       1. Setup a loop of four iterations.
                        002D    241 ;       2. Generate a stuck at 0 or 1 test pattern.
                        002D    242 ;       3. Load the literal field of a test micro-instruction with the pattern.
                        002D    243 ;       4. Load the test micro-instruction into DCS.
                        002D    244 ;       5. Execute the micro-sequence:
                        002D    245 ;               1. Write the LITERAL register to the rotator's PLATCH.
                        002D    246 ;               2. Read the PLATCH back to the RDM.
                        002D    247 ;       6. Check the results.
                        002D    248 ;       7. Loop back until done.
                        002D    249 ;
                        002D    250 ; TEST DATA
                        002D    251 ;
                        002D    252 ;       This test uses the lower six bits of the standard stuck at zero or one
                        002D    253 ;       test patterns.
                        002D    254 ;
                        002D    255 ;           Pattern Generated           Expected PLATCH contents
                        002D    256 ;
                        002D    257 ;               AAAAAAAA                        2A
                        002D    258 ;               55555555                        15
                        002D    259 ;               33333333                        33
                        002D    260 ;               0F0F0F0F                        0F
                        002D    261 ;
                        002D    262 ; LOGICAL DESCRIPTION
                        002D    263 ;
                        002D    264 ;       The data from the nine bit LITRL field is passed through the lower six
                        002D    265 ;       bits of the SBUS to the PLATCH. This involves all four SRM slices as
                        002D    266 ;       well as the SRK gate array.
                        002D    267 ;
                        002D    268 ; ASSUMPTIONS
                        002D    269 ;
                        002D    270 ;       This test requires an operational SBUS and WBUS bits <5:0>.
                        002D    271 ;       The ALP special function WMUX <- BMUX must also work.
                        002D    272 ;
                        002D    273 ; ERROR DESCRIPTION
                        002D    274 ;
                        002D    275 ;       On error this test will call out the SRM and SRK gate arrays.
                        002D    276 ;       It will also type out the following information:
                        002D    277 ;
                        002D    278 ;       EXPECTED PLATCH CONTENTS
                        002D    279 ;       RECEIVED PLATCH CONTENTS
                        002D    280 ;       LOOP COUNT
                        002D    281 ;
                        002D    282 ;--
                        002D    283 ;*****************************************************************************
                        002D    284 ;/////////////////////////////////////////////////////////////////////////////
```

```
                                  002D
                                  002D          INITIALIZE                                  ; INITIALIZE THE CPU
                                  002F          LOOP            I,1,4                       ; LOOP FOR FOUR SA01 PATTERNS
                                  0036          SA01PAT         I,1$                        ; GENERATE A SA01 TEST PATTERN
                                  003E          LDFIELD         1$,,W,2$,31,39              ; LOAD BITS <5:0> OF PATTERN TO
                                  004F          LOADDCS         2$,,1,1                     ; LITRL FIELD AND LOAD INTO DCS
                                  0056          ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                                  0058          FETCH           0                           ; SEND THE PATTERN THROUGH THE
                                  005D          BRSTCLK         3                           ; PLATCH TO THE RDM
                                  0061          CMPREGMSK       WHREG,3$,I,4$,,B            ; CHECK THE RESULT
                                  006D          IFERROR         ,<<SRM><SRK>>               ; ERROR CALLOUT
                                  0074          ENDLOOP         I                           ; CONTINUE LOOP
                                  0077          SKIP                                        ; END OF TEST
                                  007E
                                  007E    BEGIN_TEST_DATA
                                  007E
                                  007E    ;-----------------------------------------------------------------------
                                  007E
                        00000000  007E    1$:     .LONG   0       ; LOCATION TO HOLD SA01 PATTERN
                                  0082
                                  0082    2$:
U 01, 7700,1800,EF64,0355,0470,1802  0082    ;% 01   LIT/LITRL,LITRL/0AA,ROT/OLIT0.PL_LIT     ; TEST MICRO-INSTRUCTION
                                  008D
                              2A  008D    3$:     .BYTE   ^X2A    ; EXPECTED PLATCH DATA
                              15  008E            .BYTE   ^X15
                              33  008F            .BYTE   ^X33
                              0F  0090            .BYTE   ^X0F
                                  0091
                              3F  0091    4$:     .BYTE   ^X3F    ; MASK FOR DATA COMPARISON
                                  0092
                                  0092    BEGIN_CPU_DATA
                                  0002
                                  0002    DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801  0001    ;% 00   NOP                                     ; GET CPU UP TO SPEED
U 01, 7700,1800,EF64,0355,0470,1802  000C    ;% 01   LIT/LITRL,LITRL/0AA,ROT/OLIT0.PL_LIT     ; LOAD PLATCH WITH LITRL
U 02, 6700,4800,B347,0300,0470,1803  0017    ;% 02   ROT/PL,ALPCTL/WX_S.Q_0,RDM_WB            ; READ PLATCH TO RDM
U 03, 7300,C800,0364,0300,0470,1804  0022    ;% 03   STOP.CLOCK                               ; HALT
                                  002D
                                  002D    END_CPU_DATA
                                  0092
                                  0092    END_TEST_DATA
                                  0092
                                  0092    ;-----------------------------------------------------------------------
                                  0092
                                  0092    ENDTEST
```

```
                        002D        .SBTTL  "TEST   03A     Rotator Test  8 SLATCH gets Literal Test
                        002D    343 ;*****************************************************************************
                        002D    344 ;++
                        002D    345 ;
                        002D    346 ; FUNCTIONAL DESCRIPTION
                        002D    347 ;
                        002D    348 ;       This test checks the data path from the LITRL field to the SLATCH.
                        002D    349 ;
                        002D    350 ; TEST ALGORITHM
                        002D    351 ;
                        002D    352 ;       1. Setup a loop of four iterations.
                        002D    353 ;       2. Generate a stuck at 0 or 1 test pattern.
                        002D    354 ;       3. Load the literal field of a test micro-instruction with the pattern.
                        002D    355 ;       4. Load the test micro-instruction into DCS.
                        002D    356 ;       5. Execute the micro-sequence:
                        002D    357 ;               1. Write the LITERAL register to the rotator's SLATCH.
                        002D    358 ;               2. Read the SLATCH back to the RDM.
                        002D    359 ;       6. Check the results.
                        002D    360 ;       7. Loop back until done.
                        002D    361 ;
                        002D    362 ; TEST DATA
                        002D    363 ;
                        002D    364 ;       This test uses the lower six bits of the standard stuck at zero or one
                        002D    365 ;       test patterns.
                        002D    366 ;
                        002D    367 ;           Pattern Generated           Expected SLATCH contents
                        002D    368 ;
                        002D    369 ;               AAAAAAAA                        2A
                        002D    370 ;               55555555                        15
                        002D    371 ;               33333333                        33
                        002D    372 ;               0F0F0F0F                        0F
                        002D    373 ;
                        002D    374 ; LOGICAL DESCRIPTION
                        002D    375 ;
                        002D    376 ;       The data from the nine bit LITRL field is passed through the lower six
                        002D    377 ;       bits of the SBUS to the SLATCH. This involves all four SRM slices as
                        002D    378 ;       well as the SRK gate array.
                        002D    379 ;
                        002D    380 ; ASSUMPTIONS
                        002D    381 ;
                        002D    382 ;       This test requires an operational SBUS and WBUS bits <5:0>.
                        002D    383 ;       The ALP special function WMUX <- BMUX must also work.
                        002D    384 ;
                        002D    385 ; ERROR DESCRIPTION
                        002D    386 ;
                        002D    387 ;       On error this test will call out the SRM and SRK gate arrays.
                        002D    388 ;       It will also type out the following information:
                        002D    389 ;
                        002D    390 ;       EXPECTED SLATCH CONTENTS
                        002D    391 ;       RECEIVED SLATCH CONTENTS
                        002D    392 ;       LOOP COUNT
                        002D    393 ;
                        002D    394 ;--
                        002D    395 ;*****************************************************************************
                        002D    396 ;/////////////////////////////////////////////////////////////////////////////
```

```
                                  002D
                                  002D          INITIALIZE                                  ; INITIALIZE THE CPU
                                  002F          LOOP            I,1,4                       ; LOOP FOR FOUR SA01 PATTERNS
                                  0036          SA01PAT         I,1$                        ; GENERATE A SA01 TEST PATTERN
                                  003E          LDFIELD         1$,,W,2$,31,39              ; LOAD BITS <5:0> OF PATTERN TO
                                  004F          LOADDCS         2$,,1,1                     ; LITRL FIELD AND LOAD INTO DCS
                                  0056          ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                                  0058          FETCH           0                           ; SEND THE PATTERN THROUGH THE
                                  005D          BRSTCLK         3                           ; PLATCH TO THE RDM
                                  0061          CMPREGMSK       WHREG,3$,I,4$,,B            ; CHECK THE RESULT
                                  006D          IFERROR         ,<<SRM><SRK>>               ; ERROR CALLOUT
                                  0074          ENDLOOP         I                           ; CONTINUE LOOP
                                  0077          SKIP                                        ; END OF TEST
                                  007E
                                  007E    BEGIN_TEST_DATA
                                  007E
                                  007E      ;-----------------------------------------------------------------------
                                  007E
                        00000000  007E    1$:     .LONG   0       ; LOCATION TO HOLD SA01 PATTERN
                                  0082
                                  0082    2$:
U 01, 7700,1800,F764,0355,0470,1802  0082    ;% 01   LIT/LITRL,LITRL/0AA,ROT/OLIT0.SL_LIT     ; TEST MICRO-INSTRUCTION
                                  008D
                              2A  008D    3$:     .BYTE   ^X2A    ; EXPECTED SLATCH DATA
                              15  008E            .BYTE   ^X15
                              33  008F            .BYTE   ^X33
                              0F  0090            .BYTE   ^X0F
                                  0091
                              3F  0091    4$:     .BYTE   ^X3F    ; MASK FOR DATA COMPARISON
                                  0092
                                  0092    BEGIN_CPU_DATA
                                  0002
                                  0002    DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801  0001    ;% 00   NOP                                     ; GET CLOCK UP TO SPEED
U 01, 7700,1800,F764,0355,0470,1802  000C    ;% 01   LIT/LITRL,LITRL/0AA,ROT/OLIT0.SL_LIT     ; LOAD SLATCH WITH LITERAL
U 02, 6700,0800,BB47,0300,0470,1803  0017    ;% 02   ROT/SL,ALPCTL/WX_S.Q_0,RDM_WB            ; READ SLATCH TO RDM
U 03, 7300,C800,0364,0300,0470,1804  0022    ;% 03   STOP.CLOCK                               ; HALT
                                  002D
                                  002D    END_CPU_DATA
                                  0092
                                  0092    END_TEST_DATA
                                  0092
                                  0092      ;-----------------------------------------------------------------------
                                  0092
                                  0092    ENDTEST
```

```
                        001E        .SBTTL  "TEST   03B     Rotator Test  9 Zero Test
                        001E    455 ;*******************************************************************************
                        001E    456 ;++
                        001E    457 ;
                        001E    458 ; FUNCTIONAL DESCRIPTION
                        001E    459 ;
                        001E    460 ;       This test examines the rotator's zero function.
                        001E    461 ;
                        001E    462 ; TEST ALGORITHM
                        001E    463 ;
                        001E    464 ;       This test has been divided up into two subtests. The first subtest
                        001E    465 ;       examines the zero function with ones on the RBUS and MBUS, while the
                        001E    466 ;       second subtest examines the zero function with ones in the LITRL field.
                        001E    467 ;
                        001E    468 ;       Subtest 1
                        001E    469 ;
                        001E    470 ;       1. Execute the following micro-sequence:
                        001E    471 ;               1. Placing ones on the RBUS and MBUS, try to run a zero from
                        001E    472 ;                  the SBUS to the WBUS and into the RDM.
                        001E    473 ;       2. Check the result.
                        001E    474 ;
                        001E    475 ;       Subtest 2
                        001E    476 ;
                        001E    477 ;       1. Execute the following micro-sequence:
                        001E    478 ;               1. Placing ones in the LITRL field, try to run a zero from the
                        001E    479 ;                  SBUS to the WBUS and into the RDM.
                        001E    480 ;       2. Check the result.
                        001E    481 ;
                        001E    482 ; TEST DATA
                        001E    483 ;
                        001E    484 ;       Subtest 1       RBUS and MBUS Data          Expected SBUS data
                        001E    485 ;                        TEMP0 = FFFFFFFF                00000000
                        001E    486 ;
                        001E    487 ;       Subtest 2       LITRL Field Data            Expected SBUS data
                        001E    488 ;                         LITRL = 1FF                   00000000
                        001E    489 ;
                        001E    490 ; LOGICAL DESCRIPTION
                        001E    491 ;
                        001E    492 ;       The zero function is performed inside the SRM gate array. SRM gets its
                        001E    493 ;       control signals from the SRK gate array.
                        001E    494 ;
                        001E    495 ; ASSUMPTIONS
                        001E    496 ;
                        001E    497 ;       This test requires the following to be operational: the MBUS, RBUS,
                        001E    498 ;       SBUS, WBUS and the ALPCTL special function WMUX <- BMUX.
                        001E    499 ;
                        001E    500 ; ERROR DESCRIPTION
                        001E    501 ;
                        001E    502 ;       On error this test will call out the SRM and SRK gate arrays.
                        001E    503 ;       The following information will also be typed out:
                        001E    504 ;
                        001E    505 ;       EXPECTED SBUS DATA
                        001E    506 ;       RECEIVED SBUS DATA
                        001E    507 ;
                        001E    508 ;--
```

```
                                001E   ;*****************************************************************************
                                001E
                                001E           SUBTEST         1
                                0021
                                0021   ;/////////////////////////////////////////////////////////////////////////
                                0021   ;+
                                0021   ; Examine the zero function with ones on the RBUS and MBUS.
                                0021   ;-
                                0021           INITIALIZE                                  ; INITIALIZE THE CPU
                                0023           LOADDCS         2$,,1,1                     ; LOAD TEST INSTRUCTION
                                002A           ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                                002C           FETCH           0                           ; EXECUTE THE ROT/ZERO TEST
                                0031           BRSTCLK         2                           ; ...
                                0035           CMPREG          WHREG,1$,,L                 ; WAS A ZERO PUT ON THE WBUS?
                                003E           IFERROR         ,<<SRM><SRK>>               ; IF NOT, SIGNAL AN ERROR
                                0045
                                0045           SUBTEST         2
                                0048
                                0048   ;/////////////////////////////////////////////////////////////////////////
                                0048   ;+
                                0048   ; Examine the ROT/ZERO micro-order with ones in the LITRL field.
                                0048   ;-
                                0048           LOADDCS         3$,,1,1                     ; LOAD TEST INSTRUCTION
                                004F           ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                                0051           FETCH           0                           ; EXECUTE THE ROT/ZERO TEST
                                0056           BRSTCLK         2                           ; ...
                                005A           CMPREG          WHREG,1$,,L                 ; WAS A ZERO PUT ON THE WBUS?
                                0063           IFERROR         ,<<SRM><SRK>>               ; IF NOT, SIGNAL AN ERROR
                                006A           SKIP                                        ; END OF TEST
                                0071
                                0071   BEGIN_TEST_DATA
                                0071
                                0071   ;-----------------------------------------------------------------------
                                0071
                        00000000 0071   1$:     .LONG   0       ; EXPECTED SBUS DATA
                                0075
                                0075   2$:
U 01, 6700,C800,5B47,0300,0470,1802 0075   ;% 01   RSRC/TEMP0,MSRC/TEMP0,ROT/ZERO,ALPCTL/WX_S.Q_0,RDM_WB
                                0080
                                0080   3$:
U 01, 6700,D800,5B47,03FF,8470,1802 0080   ;% 01   LIT/LITRL,LITRL/1FF,ROT/ZERO,ALPCTL/WX_S.Q_0,RDM_WB
                                008B
                                008B   BEGIN_CPU_DATA
                                0002
                                0002   DCS_DATA
                                0001
U 00, 7700,C800,0364,0300,0470,1801 0001   ;% 00   NOP                                                    ; CPU CLOCK
U 01, 6700,C800,5B47,0300,0470,1802 000C   ;% 01   RSRC/TEMP0,MSRC/TEMP0,ROT/ZERO,ALPCTL/WX_S.Q_0,RDM_WB   ; TEST
U 02, 7300,4800,0364,0300,0470,1803 0017   ;% 02   STOP.CLOCK                                                ; HALT
                                0022
                                0022   RTEMP_DATA
                                0001
                        FFFFFFFF 0001           .LONG   ^XFFFFFFFF      ; INITIAL VALUE FOR RTEMP0 AND MTEMP0
                                0005
                                0005   END_CPU_DATA
```

```
                        008B   573
                        008B   574 END_TEST_DATA
                        008B
                        008B       ;-----------------------------------------------------------------------------
                        008B   575
                        008B   576 ENDTEST
```

```
0023          .SBTTL  "TEST   03C     Rotator Test 10 Minus One Test
0023    578 ;*****************************************************************************
0023    579 ;++
0023    580 ;
0023    581 ; FUNCTIONAL DESCRIPTION
0023    582 ;
0023    583 ;       This test examines the rotator's minus one function.
0023    584 ;
0023    585 ; TEST ALGORITHM
0023    586 ;
0023    587 ;       This test has been divided up into two subtests. The first subtest
0023    588 ;       examines the minus one function with zeros on the RBUS and MBUS, while
0023    589 ;       the second subtest examines the minus one function with zeros in the
0023    590 ;       LITRL field.
0023    591 ;
0023    592 ;       Subtest 1
0023    593 ;
0023    594 ;       1. Execute the following micro-sequence:
0023    595 ;               1. Placing zeros on the RBUS and MBUS, try to run minus one
0023    596 ;                  from the SBUS to the WBUS and into the RDM.
0023    597 ;       2. Check the result.
0023    598 ;
0023    599 ;       Subtest 2
0023    600 ;
0023    601 ;       1. Execute the following micro-sequence:
0023    602 ;               1. Placing zeros in the LITRL field, try to run minus one from
0023    603 ;                  the SBUS to the WBUS and into the RDM.
0023    604 ;       2. Check the result.
0023    605 ;
0023    606 ; TEST DATA
0023    607 ;
0023    608 ;       Subtest 1       RBUS and MBUS Data          Expected SBUS data
0023    609 ;                        TEMP0 = 00000000               FFFFFFFF
0023    610 ;
0023    611 ;       Subtest 2       LITRL Field Data            Expected SBUS data
0023    612 ;                         LITRL = 000                  FFFFFFFF
0023    613 ;
0023    614 ; LOGICAL DESCRIPTION
0023    615 ;
0023    616 ;       The minus one function is performed inside the SRM gate array. SRM gets
0023    617 ;       control signals from the SRK gate array.
0023    618 ;
0023    619 ; ASSUMPTIONS
0023    620 ;
0023    621 ;       This test requires the following to be operational: the MBUS, RBUS,
0023    622 ;       SBUS, WBUS and the ALPCTL special function WMUX <- BMUX.
0023    623 ;
0023    624 ; ERROR DESCRIPTION
0023    625 ;
0023    626 ;       On error this test will call out the SRM and SRK gate arrays.
0023    627 ;       The following information will also be typed out:
0023    628 ;
0023    629 ;       EXPECTED SBUS DATA
0023    630 ;       RECEIVED SBUS DATA
0023    631 ;
```

```
                        0023   632 ;--
                        0023   633 ;******************************************************************************
                        0023   634
                        0023   635         SUBTEST         1
                        0026
                        0026       ;////////////////////////////////////////////////////////////////////////////
                        0026   636 ;+
                        0026   637 ; Examine the ROT/MINUS1 micro-order with zeros on the RBUS and MBUS.
                        0026   638 ;-
                        0026   639         INITIALIZE                              ; INITIALIZE THE CPU
                        0028   640         LOADDCS         2$,,1,1                 ; LOAD TEST INSTRUCTION
                        002F   641         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        0031   642         FETCH           0                       ; EXECUTE THE ROT/MINUS1
                        0036   643         BRSTCLK         2                       ; MICRO-ORDER
                        003A   644         CMPREG          WHREG,1$,,L             ; WAS A MINUS1 PUT ON THE WBUS?
                        0043   645         IFERROR         ,<<SRM><SRK>>           ; IF NOT, SIGNAL AN ERROR
                        004A   646
                        004A   647         SUBTEST         2
                        004D
                        004D       ;////////////////////////////////////////////////////////////////////////////
                        004D   648 ;+
                        004D   649 ; Examine the ROT/MINUS1 micro-order with zeros in the LITRL field.
                        004D   650 ;-
                        004D   651         LOADDCS         3$,,1,1                 ; LOAD TEST INSTRUCTION
                        0054   652         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        0056   653         FETCH           0                       ; EXECUTE THE ROT/MINUS1
                        005B   654         BRSTCLK         2                       ; MICRO-ORDER
                        005F   655         CMPREG          WHREG,1$,,L             ; WAS A MINUS1 PUT ON THE WBUS?
                        0068   656         IFERROR         ,<<SRM><SRK>>           ; IF NOT, SIGNAL AN ERROR
                        006F   657         SKIP                                    ; END OF TEST
                        0076   658
                        0076   659 BEGIN_TEST_DATA
                        0076
                        0076       ;-------------------------------------------------------------------------
                        0076   660
                FFFFFFFF  0076   661 1$:     .LONG   ^XFFFFFFFF      ; EXPECTED WBUS DATA
                        007A   662
                        007A   663 2$:
U 01, 6700,8800,E747,0300,0470,1802  007A   664 ;% 01   RSRC/TEMP0,MSRC/TEMP0,ROT/MINUS1,ALPCTL/WX_S.Q_0,RDM_WB
                        0085   668
                        0085   669 3$:
U 01, 6700,1800,E747,0300,0470,1802  0085   670 ;% 01   LIT/LITRL,LITRL/000,ROT/MINUS1,ALPCTL/WX_S.Q_0,RDM_WB
                        0090   674
                        0090   675 BEGIN_CPU_DATA
                        0002   676
                        0002   677 DCS_DATA
                        0001   678
U 00, 7700,C800,0364,0300,0470,1801  0001   679 ;% 00   NOP                                                    ; NOP FOR CPU
U 01, 6700,8800,E747,0300,0470,1802  000C   683 ;% 01   RSRC/TEMP0,MSRC/TEMP0,ROT/MINUS1,ALPCTL/WX_S.Q_0,RDM_WB ; TEST
U 02, 7300,4800,0364,0300,0470,1803  0017   687 ;% 02   STOP.CLOCK                                             ; HALT
                        0022   691
                        0022   692 RTEMP_DATA
                        0001   693
                00000000  0001   694         .LONG   0       ; VALUE FOR RTEMP0 AND MTEMP0
                        0005   695
```

```
                0005   696 END_CPU_DATA
                0090   697
                0090   698 END_TEST_DATA
                0090
                0090       ;-----------------------------------------------------------------------------
                0090   699
                0090   700 ENDTEST
```

```
0029          .SBTTL  "TEST   03D     Rotator Test 11 DSIZE Constants Test
0029    702 ;*****************************************************************************
0029    703 ;++
0029    704 ;
0029    705 ; FUNCTIONAL DESCRIPTION
0029    706 ;
0029    707 ;       This test examines the rotator's ability to generate a zero extended
0029    708 ;       constant of 1, 2, or 4 as a function of DSIZE. For this test DSIZE is
0029    709 ;       determined by the DTYPE field.
0029    710 ;
0029    711 ;       This test can only do checks for byte, word, and longword data types.
0029    712 ;       Quadword data types must wait until after the test of the DSIZE ROM.
0029    713 ;
0029    714 ; TEST ALGORITHM
0029    715 ;
0029    716 ;       This test has been divided up into two subtests. The first subtest
0029    717 ;       examines the DSIZE constant function with ones on the RBUS and MBUS,
0029    718 ;       while the second subtest examines the DSIZE constant function with ones
0029    719 ;       in the LITRL field.
0029    720 ;
0029    721 ;       Subtest 1
0029    722 ;
0029    723 ;       1. Setup a loop of three iterations.
0029    724 ;       2. Load the DTYPE field of a test micro-instruction for a byte, word or
0029    725 ;          longword data type.
0029    726 ;       3. Load the test micro-instruction into DCS.
0029    727 ;       4. Execute the following micro-sequence:
0029    728 ;             1. Placing ones on the RBUS and MBUS, try to run the DSIZE
0029    729 ;                constant from the SBUS to the WBUS and into the RDM.
0029    730 ;       5. Check the result.
0029    731 ;       6. Loop until done.
0029    732 ;
0029    733 ;       Subtest 2
0029    734 ;
0029    735 ;       1. Setup a loop of three iterations.
0029    736 ;       2. Load the DTYPE field of a test micro-instruction for a byte, word or
0029    737 ;          longword data type.
0029    738 ;       3. Load the test micro-instruction into DCS.
0029    739 ;       4. Execute the following micro-sequence:
0029    740 ;             1. Placing ones in the LITRL field, try to run the DSIZE
0029    741 ;                constant from the SBUS to the WBUS and into the RDM.
0029    742 ;       5. Check the result.
0029    743 ;       6. Loop until done.
0029    744 ;
0029    745 ; TEST DATA
0029    746 ;
0029    747 ;       Subtest 1       Loop        RBUS and MBUS Data      Expected SBUS Data
0029    748 ;                         1.              FFFFFFFF                00000001
0029    749 ;                         2.              FFFFFFFF                00000002
0029    750 ;                         3.              FFFFFFFF                00000004
0029    751 ;
0029    752 ;       Subtest 2       Loop         LITRL Field Data       Expected SBUS Data
0029    753 ;                         1.                  1FF                 00000001
0029    754 ;                         2.                  1FF                 00000002
0029    755 ;                         3.                  1FF                 00000004
```

```
                        0029    756 ;
                        0029    757 ; LOGICAL DESCRIPTION
                        0029    758 ;
                        0029    759 ;       DSIZE constants are generated inside the SRM gate array. SRM gets its
                        0029    760 ;       control signals from the SRK gate array which decodes the DSIZE field.
                        0029    761 ;
                        0029    762 ; ASSUMPTIONS
                        0029    763 ;
                        0029    764 ;       This test requires an operational MBUS, RBUS, SBUS and WBUS. The ALPCTL
                        0029    765 ;       special function WMUX <- BMUX must work as well as the multiplexers
                        0029    766 ;       that gate in the DSIZE field of the micro-word.
                        0029    767 ;
                        0029    768 ; ERROR DESCRIPTION
                        0029    769 ;
                        0029    770 ;       On error this test will call out the SRM and SRK gate arrays.
                        0029    771 ;       The following information will also be typed:
                        0029    772 ;
                        0029    773 ;       EXPECTED SBUS DATA
                        0029    774 ;       RECEIVED SBUS DATA
                        0029    775 ;       LOOP COUNT
                        0029    776 ;
                        0029    777 ;--
                        0029    778 ;*****************************************************************************
                        0029    779
                        0029    780         SUBTEST         1
                        002C
                        002C        ;/////////////////////////////////////////////////////////////////////////////
                        002C    781 ;+
                        002C    782 ; Examine the DSIZE constants with ones on the RBUS and MBUS.
                        002C    783 ;-
                        002C    784         INITIALIZE                              ; INTIALIZE THE CPU
                        002E    785         LOOP            I,1,3                   ; LOOP FOR 3 DATA TYPES
                        0035    786         LDFIELD         3$,I,B,1$,40,41         ; LOAD DTYPE DATA
                        0046    787         LOADDCS         1$,,1,1                 ; LOAD TEST MICRO-INSTRUCTION
                        004D    788         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        004F    789         FETCH           0                       ; EXECUTE TEST INSTRUCTION
                        0054    790         BRSTCLK         2                       ; ...
                        0058    791         CMPREG          WHREG,4$,I,L            ; WAS THE WBUS DATA CORRECT?
                        0061    792         IFERROR         ,<<SRM><SRK>>           ; IF NOT, SIGNAL AN ERROR
                        0068    793         ENDLOOP         I                       ; CONTINUE LOOP
                        006B    794
                        006B    795         SUBTEST         2
                        006E
                        006E        ;/////////////////////////////////////////////////////////////////////////////
                        006E    796 ;+
                        006E    797 ; Examine the DSIZE constants with ones in the LITRL field.
                        006E    798 ;-
                        006E    799         LOOP            I,1,3                   ; LOOP FOR 3 DATA TYPES
                        0075    800         LDFIELD         3$,I,B,2$,40,41         ; LOAD DTYPE DATA
                        0086    801         LOADDCS         2$,,1,1                 ; LOAD TEST MICRO-INSTRUCTION
                        008D    802         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        008F    803         FETCH           0                       ; EXECUTE TEST INSTRUCTION
                        0094    804         BRSTCLK         2                       ; ...
                        0098    805         CMPREG          WHREG,4$,I,L            ; WAS THE WBUS DATA CORRECT?
                        00A1    806         IFERROR         ,<<SRM><SRK>>           ; IF NOT, SIGNAL AN ERROR
```

```
                                  00A8          ENDLOOP        I                          ; CONTINUE LOOP
                                  00AB          SKIP                                      ; END OF TEST
                                  00B2
                                  00B2  BEGIN_TEST_DATA
                                  00B2
                                  00B2      ;-----------------------------------------------------------------------
                                  00B2
                                  00B2  1$:
U 01, 6701,C800,7347,0000,0470,1802  00B2  ;% 01   RSRC/TEMP0,MSRC/TEMP0,DTYPE/BYTE,ROT/CONX.SIZ,ALPCTL/WX_S.Q_0,RDM_WB,VSIZE/1
                                  00BD
                                  00BD  2$:
U 01, 6701,D800,7347,00FF,8470,1802  00BD  ;% 01   LIT/LITRL,LITRL/1FF,DTYPE/BYTE,ROT/CONX.SIZ,ALPCTL/WX_S.Q_0,RDM_WB,VSIZE/1
                                  00C8
                              00  00C8  3$:     .BYTE   0       ; DTYPE = BYTE
                              01  00C9          .BYTE   1       ; DTYPE = WORD
                              02  00CA          .BYTE   2       ; DTYPE = LONGWORD
                                  00CB
                        00000001  00CB  4$:     .LONG   ^X00000001      ; SBUS CONSTANT FOR DTYPE = BYTE
                        00000002  00CF          .LONG   ^X00000002      ; SBUS CONSTANT FOR DTYPE = WORD
                        00000004  00D3          .LONG   ^X00000004      ; SBUS CONSTANT FOR DTYPE = LONGWORD
                                  00D7
                                  00D7  BEGIN_CPU_DATA
                                  0002
                                  0002  DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 6701,C800,7347,0000,0470,1802  000C  ;% 01   RSRC/TEMP0,MSRC/TEMP0,DTYPE/BYTE,ROT/CONX.SIZ,ALPCTL/WX_S.Q_0,RDM_WB,VSIZE/1
U 02, 7300,4800,0364,0300,0470,1803  0017  ;% 02   STOP.CLOCK
                                  0022
                                  0022  RTEMP_DATA
                                  0001
                        FFFFFFFF  0001          .LONG   ^XFFFFFFFF      ; ONES FOR RTEMP0 AND MTEMP0
                                  0005
                                  0005  END_CPU_DATA
                                  00D7
                                  00D7  END_TEST_DATA
                                  00D7
                                  00D7      ;-----------------------------------------------------------------------
                                  00D7
                                  00D7  ENDTEST
```

```
                        003F        .SBTTL  "TEST   03E     Rotator Test 12 Extract MBUS concatenated with RBUS Test 1
                        003F    859 ;*******************************************************************************
                        003F    860 ;++
                        003F    861 ;
                        003F    862 ;  FUNCTIONAL DESCRIPTION
                        003F    863 ;
                        003F    864 ;       This test checks that the Super Rotator will extract a bit field from
                        003F    865 ;       MBUS'RBUS. The starting bit position is specified by the Position Latch
                        003F    866 ;       (PLATCH) which will be incremented from 0 to 31. The size of the
                        003F    867 ;       field is specified by the Size Latch (SLATCH) which will be set to 1.
                        003F    868 ;       The bit that is to be extracted will be preceded by a 0 in a field of
                        003F    869 ;       1's.
                        003F    870 ;
                        003F    871 ;  TEST ALGORITHM
                        003F    872 ;
                        003F    873 ;       1. Load the micro words to perform the test.
                        003F    874 ;       2. Setup a loop of 31 iterations.
                        003F    875 ;       3. Select and load values from the tables.
                        003F    876 ;       4. Execute the micro sequence.
                        003F    877 ;               1. Set Size Latch (SLATCH) to 1.
                        003F    878 ;               2. Increment Position Latch (PLATCH).
                        003F    879 ;               3. Extract and Zero Extend Data and put it on the WBUS.
                        003F    880 ;       5. Check the results of the WBUS read.
                        003F    881 ;       6. Looped 32 times? If not go to step 3.
                        003F    882 ;
                        003F    883 ;  DATA PATTERNS
                        003F    884 ;
                        003F    885 ;       FFFFFFFF
                        003F    886 ;             01
                        003F    887 ;
                        003F    888 ;  LOGIC DESCRIPTION
                        003F    889 ;
                        003F    890 ;       These subtests check the gates that effect the Extract and Zero Extend
                        003F    891 ;       function in the SRM and SRK gate arrays.
                        003F    892 ;
                        003F    893 ;  ASSUMPTIONS
                        003F    894 ;
                        003F    895 ;       This test requires no signals from any other CPU Module but assumes
                        003F    896 ;       an operational RBUS, MBUS, SBUS, WBUS and ALPCTL special function
                        003F    897 ;       WMUX <- BMUX.
                        003F    898 ;
                        003F    899 ;  ERROR DESCRIPTION
                        003F    900 ;
                        003F    901 ;       On error this test will type out the SRM and SRK gate arrays.
                        003F    902 ;       The following information will also be type out:
                        003F    903 ;
                        003F    904 ;       EXPECTED DATA
                        003F    905 ;       RECEIVED DATA
                        003F    906 ;       LOOP COUNT
                        003F    907 ;       RTEMPS 0-4
                        003F    908 ;
                        003F    909 ;--
                        003F    910 ;*******************************************************************************
                        003F    911 ;///////////////////////////////////////////////////////////////////////////////
                        003F    912
```

```
                                  003F          INITIALIZE                                  ;DCLO low,Init Scratchpads
                                  0041          LDFIELD        40$,,L,2$,00,31,,,,,NS       ;Initialize 2$
                                  0052          LOADDCS        1$,,0,7                      ;Load the test micro instructions
                                  0059          LOOP           I,1,31                       ;Set loop count
                                  0060          SAVEINDEX      I,20$                        ;Save loop count at 20$
                                  0065          LDFIELD        20$,,L,30$,31,62,LL          ;Load LONLIT field
                                  0076          LOADDCS        30$,,1,1                     ;Load LONLIT micro word
                                  007D          LDFIELD        3$,,B,2$,4$,5$,,I,I,,NS      ;Load RBUS data
                                  008E          LOADREG        WDREG,2$,,L                  ;Load RDM D Register
                                  0096          ERRLOOP                                     ;Error Loop
                                  0098          FETCH          0                            ;Fetch the first micro word
                                  009D          BRSTCLK        6                            ;Clock stop location 6
                                  00A1          CMPREG         WHREG,10$,,L                 ;Check the received data
                                  00AA          IFERROR        4,<<SRM><SRK>>               ;Continue if no error else call
                                  00B1          ENDLOOP        I                            ;Continue with loop to increment PLA
                                  00B4          SKIP                                        ;End of test
                                  00BB
                                  00BB    BEGIN_TEST_DATA
                                  00BB
                                  00BB    ;-----------------------------------------------------------------------
                                  00BB
                                  00BB    1$:
U 00, 7700,C800,0364,0300,0470,1801  00BB    ;% 00   NOP                                                 ;NOP Micro Instruction
                                  00C6    30$:
U 01, 7700,7800,7FFF,FFFF,8470,1802  00C6    ;% 01   LONLIT_[0]                                          ;Setup to load PLATCH
U 02, 7700,4800,B447,0310,0470,1803  00D1    ;% 02   ROT/PL.SL_WB,RSRC/TEMP4,ALPCTL/WX_R.Q_M              ;Set SLATCH
U 03, 7700,C800,BC47,03D4,0470,1804  00DC    ;% 03   ROT/SL.PL_WB,RSRC/LONLIT,ALPCTL/WX_R.Q_M             ;Set PLATCH
U 04, 5700,0840,0364,030C,0470,1805  00E7    ;% 04   R[TEMP3]_RDM                                        ;RBUS of MBUS'RBUS
U 05, 6700,0802,0370,030C,0470,1806  00F2    ;% 05   RDM_WB,ROT/XZ.MR,ALPCTL/WX_S,MSRC/TEMP2,RSRC/TEMP3 ;Test word
U 06, 7300,C800,0364,0300,0470,1807  00FD    ;% 06   NOP,STOP.CLOCK                                      ;Stop clock
                                  0108
                         FFFFFFFF  0108    2$:     .Long   ^xFFFFFFFF                  ;RBUS data
                                  010C
                               01  010C    3$:     .Byte   1                           ;RBUS data
                                  010D
                               00  010D    4$:     .Byte   0                           ;PLATCH data
                                  010E
          08 07 06 05 04 03 02 01  010E    5$:     .Byte   01,02,03,04,05,06,07,08     ;PLATCH data
          10 0F 0E 0D 0C 0B 0A 09  0116            .Byte   09,10,11,12,13,14,15,16     ;PLATCH data
          18 17 16 15 14 13 12 11  011E            .Byte   17,18,19,20,21,22,23,24     ;PLATCH data
             1F 1E 1D 1C 1B 1A 19  0126            .Byte   25,26,27,28,29,30,31        ;PLATCH data
                                  012D
                         00000001  012D    10$:    .Long   1                           ;Check data
                                  0131
                         00000000  0131    20$:    .Long   0                           ;Index saved here
                                  0135
                         FFFFFFFF  0135    40$:    .Long   ^xFFFFFFFF                  ;Initialize 2$
                                  0139
                                  0139    BEGIN_CPU_DATA
                                  0002
                                  0002    RTEMP_DATA
                                  0001
                         00000000  0001            .Long   0                           ;RTEMP0
                         00000000  0005            .Long   0                           ;RTEMP1
                         FFFFFFFF  0009            .Long   ^xFFFFFFFF                  ;RTEMP2
```

```
00000000  000D   987          .Long   0                       ;RTEMP3
00000001  0011   988          .Long   1                       ;RTEMP4 SLATCH data
          0015   989
          0015   990 END_CPU_DATA
          0139   991
          0139   992 END_TEST_DATA
          0139
          0139          ;-------------------------------------------------------------------------------
          0139   993
          0139   994 ENDTEST
```

```
                        003F        .SBTTL  "TEST   03F     Rotator Test 13 Extract MBUS concatenated with RBUS Test 2
                        003F    996 ;*******************************************************************************
                        003F    997 ;++
                        003F    998 ;
                        003F    999 ;  FUNCTIONAL DESCRIPTION
                        003F   1000 ;
                        003F   1001 ;         This test checks that the Super Rotator will extract a bit field from
                        003F   1002 ;       MBUS'RBUS.The STARTING BIT POSITION  is specified by the Position Latch
                        003F   1003 ;       (PLATCH) which will be incremented from 0 to 31. The  SIZE of the field
                        003F   1004 ;       is specified by the Size Latch (SLATCH) which will be incremented  from
                        003F   1005 ;       1 to 31. For each increment of the  Size  Latch  (SLATCH)  the Position
                        003F   1006 ;       Latch (PLATCH) will be incremented 32 times thus rotating the  STARTING
                        003F   1007 ;       BIT POSITION  across the MBUS'RBUS from RBUS<00> to RBUS<31>.Increment-
                        003F   1008 ;       ing the Size Latch (SLATCH) will select a field width from 1 to 32 bits
                        003F   1009 ;
                        003F   1010 ;  TEST ALGORITHM
                        003F   1011 ;
                        003F   1012 ;       1. Load the micro words to perform the test.
                        003F   1013 ;       2. Setup a loop of 29 iterations.
                        003F   1014 ;       3. Select and load values from the tables.
                        003F   1015 ;       4. Setup another loop of 31 iterations.
                        003F   1016 ;       5. Select and load values from the tables.
                        003F   1017 ;       6. Execute the micro sequence.
                        003F   1018 ;                1. Increment Size Latch (SLATCH).
                        003F   1019 ;                2. Increment Position Latch (PLATCH).
                        003F   1020 ;                3. Extract and Zero Extend data onto WBUS.
                        003F   1021 ;       7. Check the results of the WBUS read.
                        003F   1022 ;       8. Looped 32 times? If not go to step five.
                        003F   1023 ;       9. Looped 29 times? If not go to step three.
                        003F   1024 ;
                        003F   1025 ;  DATA PATTERNS
                        003F   1026 ;
                        003F   1027 ;       FFFFFFFF
                        003F   1028 ;             01
                        003F   1029 ;
                        003F   1030 ;  LOGIC DESCRIPTION
                        003F   1031 ;
                        003F   1032 ;       These subtests check the gates that effect the Extract and Zero Extend
                        003F   1033 ;       function.
                        003F   1034 ;
                        003F   1035 ;  ASSUMPTIONS
                        003F   1036 ;
                        003F   1037 ;       This test requires no signals from any other CPU Module but assumes
                        003F   1038 ;       an operational RBUS, MBUS, SBUS and WBUS.
                        003F   1039 ;
                        003F   1040 ;  ERROR DESCRIPTION
                        003F   1041 ;
                        003F   1042 ;       EXPTECTED DATA
                        003F   1043 ;       RECEIVED DATA
                        003F   1044 ;       LOOP COUNT I
                        003F   1045 ;       LOOP COUNT J
                        003F   1046 ;       RTEMPS 0-4
                        003F   1047 ;
                        003F   1048 ;--
                        003F   1049 ;*******************************************************************************
```

```
                                003F  ;////////////////////////////////////////////////////////////////////////////
                                003F
                                003F          INITIALIZE                                  ;DCLO low,init Scratchpads
                                0041          LOOP            J,3,32                      ;Set loop count
                                0048          SAVEINDEX       J,50$                       ;Save loop count for SLATCH
                                004D          LDFIELD         40$,,L,2$,00,31,,,,,NS      ;Initialize 2$
                                005E          LDFIELD         40$,,L,22$,00,31,,,,,NS     ;Initialize 22$
                                006F          LOADDCS         1$,,0,<^xA>                 ;Load the test micro instructions
                                0076          LDFIELD         50$,,L,60$,31,62,LL         ;Load LONLIT field
                                0087          LOADDCS         60$,,3,1                    ;Load LONLIT micro word
                                008E          LOOP            I,1,32                      ;Set loop count
                                0095          SAVEINDEX       I,20$                       ;Save loop count for PLATCH
                                009A          LDFIELD         20$,,L,30$,31,62,LL         ;Load LONLIT field
                                00AB          LOADDCS         30$,,1,1                    ;Load LONLIT micro word
                                00B2          LDFIELD         3$,,B,2$,4$,5$,,I,I,,NS     ;Load RBUS data
                                00C3          LDFIELD         22$,,L,70$,31,62,LL         ;Load LONLIT field
                                00D4          LOADDCS         70$,,6,1                    ;Load LONLIT micro word
                                00DB          LOADREG         WDREG,2$,,L                 ;Load RDM D Register
                                00E3          ERRLOOP                                     ;Error Loop
                                00E5          FETCH           0                           ;Get first micro word
                                00EA          BRSTCLK         9                           ;Clock stop location 9
                                00EE          CMPREG          WHREG,10$,J,L               ;Check the received results
                                00F7          IFERROR         4,<<SRM><SRK>>              ;Continue if no error else call erro
                                00FE          ENDLOOP         I                           ;Continue with loop to increment PLA
                                0101          ENDLOOP         J                           ;Continue with loop to increment SLA
                                0104          SKIP                                        ;End of test
                                010B
                                010B  BEGIN_TEST_DATA
                                010B
                                010B          ;-----------------------------------------------------------------------
                                010B
                                010B  1$:
U 00, 7700,C800,0364,0300,0470,1801  010B  ;% 00   NOP                                          ;NOP Micro Instruction
                                0116  30$:
U 01, 7700,7800,7FFF,FFFF,8470,1802  0116  ;% 01   LONLIT_[0]                                   ;PLATCH data
U 02, 7700,4800,BC47,03D4,0470,1803  0121  ;% 02   ROT/SL.PL_WB,RSRC/LONLIT,ALPCTL/WX_R.Q_M     ;Set PLATCH
                                012C  60$:
U 03, 7700,7800,7FFF,FFFF,8470,1804  012C  ;% 03   LONLIT_[0]                                   ;SLATCH data
U 04, 7700,0800,B447,03D4,0470,1805  0137  ;% 04   ROT/PL.SL_WB,RSRC/LONLIT,ALPCTL/WX_R.Q_M     ;Set SLATCH
U 05, 5700,0840,0364,030C,0470,1806  0142  ;% 05   R[TEMP3]_RDM                                 ;RBUS data
                                014D  70$:
U 06, 7700,7800,7FFF,FFFF,8470,1807  014D  ;% 06   LONLIT_[0]                                   ;MBUS data
U 07, 7700,8868,0247,03D4,0470,1808  0158  ;% 07   M[TEMP8]_LONLIT
U 08, 6700,0808,0370,030C,0470,1809  0163  ;% 08   RDM_WB,ROT/XZ.MR,ALPCTL/WX_S,MSRC/TEMP8,RSRC/TEMP3 ;Test micro word
U 09, 7300,4800,0364,0300,0470,180A  016E  ;% 09   NOP,STOP.CLOCK                              ;Stop clock
                                0179
                     FFFFFFFF   0179  2$:     .Long   ^xFFFFFFFF              ;RBUS data
                                017D
                     FFFFFFFF   017D  22$:    .Long   ^xFFFFFFFF              ;RBUS data
                                0181
                           01   0181  3$:     .Byte   1                       ;PLATCH data
                                0182
                           00   0182  4$:     .Byte   0                       ;PLATCH data
                                0183
          08 07 06 05 04 03 02 01  0183  5$:     .Byte   01,02,03,04,05,06,07,08 ;PLATCH data
```

```
                10 0F 0E 0D 0C 0B 0A 09  018B  1133          .Byte   09,10,11,12,13,14,15,16 ;PLATCH data
                18 17 16 15 14 13 12 11  0193  1134          .Byte   17,18,19,20,21,22,23,24 ;PLATCH data
                20 1F 1E 1D 1C 1B 1A 19  019B  1135          .Byte   25,26,27,28,29,30,31,32 ;PLATCH data
                                         01A3  1136 ;
                                         01A3  1137 ; Expected Data
                                         01A3  1138 ;
00000005 00000001 00000000 00000000      01A3  1139 10$:      .Long   ^x00000000,^x00000000,^x00000001,^x00000005,^x0000000D
                                   0000000D  01B3
000000FD 0000007D 0000003D 0000001D      01B7  1140          .Long   ^x0000001D,^x0000003D,^x0000007D,^x000000FD,^x000001FD
                                   000001FD  01C7
00001FFD 00000FFD 000007FD 000003FD      01CB  1141          .Long   ^x000003FD,^x000007FD,^x00000FFD,^x00001FFD,^x00003FFD
                                   00003FFD  01DB
0003FFFD 0001FFFD 0000FFFD 00007FFD      01DF  1142          .Long   ^x00007FFD,^x0000FFFD,^x0001FFFD,^x0003FFFD,^x0007FFFD
                                   0007FFFD  01EF
007FFFFD 003FFFFD 001FFFFD 000FFFFD      01F3  1143          .Long   ^x000FFFFD,^x001FFFFD,^x003FFFFD,^x007FFFFD,^x00FFFFFD
                                   00FFFFFD  0203
0FFFFFFD 07FFFFFD 03FFFFFD 01FFFFFD      0207  1144          .Long   ^x01FFFFFD,^x03FFFFFD,^x07FFFFFD,^x0FFFFFFD,^x1FFFFFFD
                                   1FFFFFFD  0217
         FFFFFFFD 7FFFFFFD 3FFFFFFD      021B  1145          .Long   ^x3FFFFFFD,^x7FFFFFFD,^xFFFFFFFD
                                         0227  1146
                                   00000000  0227  1147 20$:      .Long   0                       ;Index to increment PLATCH saved here
                                         022B  1148
                                   FFFFFFFF  022B  1149 40$:      .Long   ^xFFFFFFFF              ;Data to initialize 2$
                                         022F  1150
                                   00000000  022F  1151 50$:      .Long   0                       ;Index to increment SLATCH saved here
                                         0233  1152
                                         0233  1153 BEGIN_CPU_DATA
                                         0002  1154
                                         0002  1155 RTEMP_DATA
                                         0001  1156
                                   00000000  0001  1157          .Long   0                       ;RTEMP0
                                   00000000  0005  1158          .Long   0                       ;RTEMP1
                                   FFFFFFFF  0009  1159          .Long   ^xFFFFFFFF              ;RTEMP2
                                   00000000  000D  1160          .Long   0                       ;RTEMP3
                                         0011  1161
                                         0011  1162 END_CPU_DATA
                                         0233  1163
                                         0233  1164 END_TEST_DATA
                                         0233
                                         0233       ;-----------------------------------------------------------------------------
                                         0233  1165
                                         0233  1166 ENDTEST
```

```
                        003D        .SBTTL  "TEST   040     Rotator Test 14 Extract MBUS concatenated with MBUS Test
                        003D  1168 ;*******************************************************************************
                        003D  1169 ;++
                        003D  1170 ;
                        003D  1171 ;  FUNCTIONAL DESCRIPTION
                        003D  1172 ;
                        003D  1173 ;         This test checks that the Super Rotator will extract a bit field from
                        003D  1174 ;       MBUS'MBUS.The STARTING BIT POSITION is specified by the Position  Latch
                        003D  1175 ;       (PLATCH) which will be incremented from 0 to 31. The  SIZE of the field
                        003D  1176 ;       is specified by the Size Latch (SLATCH) which will be incremented  from
                        003D  1177 ;       1 to 31. For each increment of the  Size  Latch  (SLATCH)  the Position
                        003D  1178 ;       Latch  (PLATCH) will be incremented 32 times thus rotating the STARTING
                        003D  1179 ;       BIT POSITION across the MBUS'MBUS from MBUS<00> to MBUS<31>. Increment-
                        003D  1180 ;       ing the Size Latch (SLATCH) will select a field width from 1 to 32 bits
                        003D  1181 ;
                        003D  1182 ;  TEST ALGORITHM
                        003D  1183 ;
                        003D  1184 ;       1. Load the micro words to perform the test.
                        003D  1185 ;       2. Setup a loop of 29 iterations.
                        003D  1186 ;       3. Select and load values from the tables.
                        003D  1187 ;       4. Setup another loop of 31 iterations.
                        003D  1188 ;       5. Select and load values from the tables.
                        003D  1189 ;       6. Execute the micro sequence.
                        003D  1190 ;                1. Increment Size Latch (SLATCH).
                        003D  1191 ;                2. Increment Position Latch (PLATCH).
                        003D  1192 ;                3. Extract and Zero Extend data onto WBUS.
                        003D  1193 ;       7. Check the results of the WBUS read.
                        003D  1194 ;       8. Looped 32 times? If not go to step five.
                        003D  1195 ;       9. Looped 29 times? If not go to step three.
                        003D  1196 ;
                        003D  1197 ;  DATA PATTERNS
                        003D  1198 ;
                        003D  1199 ;       FFFFFFFF
                        003D  1200 ;             01
                        003D  1201 ;
                        003D  1202 ;  LOGIC DESCRIPTION
                        003D  1203 ;
                        003D  1204 ;       These subtests check the gates that effect the Extract and Zero Extend
                        003D  1205 ;       function.
                        003D  1206 ;
                        003D  1207 ;  ASSUMPTIONS
                        003D  1208 ;
                        003D  1209 ;       This test requires no signals from any other CPU Module but assumes
                        003D  1210 ;       an operational RBUS, MBUS, SBUS and WBUS.
                        003D  1211 ;
                        003D  1212 ;  ERROR DESCRIPTION
                        003D  1213 ;
                        003D  1214 ;       EXPECTED DATA (SEE 10$)
                        003D  1215 ;       RECEIVED DATA
                        003D  1216 ;       LOOP COUNT I
                        003D  1217 ;       LOOP COUNT J
                        003D  1218 ;       RTEMPS 0-4
                        003D  1219 ;
                        003D  1220 ;--
                        003D  1221 ;*******************************************************************************
```

```
                                003D  1222 ;////////////////////////////////////////////////////////////////////////////////
                                003D  1223
                                003D  1224          INITIALIZE                                      ;DCLO low,init Scratchpads
                                003F  1225          LOOP            J,3,32                          ;Set loop count
                                0046  1226          SAVEINDEX       J,50$                           ;Save loop count for SLATCH
                                004B  1227          LDFIELD         40$,,L,2$,00,31,,,,,NS          ;Initialize 2$
                                005C  1228          LDFIELD         40$,,L,22$,00,31,,,,,NS         ;Initialize 22$
                                006D  1229          LOADDCS         1$,,0,<^xA>                     ;Load test micro words
                                0074  1230          LDFIELD         50$,,Long,60$,31,62,LL          ;Load LONLIT field
                                0085  1231          LOADDCS         60$,,3,1                        ;Load LONLIT micro word
                                008C  1232          LOOP            I,1,32                          ;Set loop count
                                0093  1233          SAVEINDEX       I,20$                           ;Save loop count for PLATCH
                                0098  1234          LDFIELD         20$,,L,30$,31,62,LL             ;Load LONLIT field
                                00A9  1235          LOADDCS         30$,,1,1                        ;Load LONLIT micro word
                                00B0  1236          LDFIELD         99$,I,B,2$,0,7,,,,,NS           ;Load LSByte of 99$
                                00C1  1237          LDFIELD         3$,,B,2$,4$,5$,,I,I,,NS         ;Load PLATCH data
                                00D2  1238          LDFIELD         22$,,L,70$,31,62,LL             ;Load LONLIT field
                                00E3  1239          LOADDCS         70$,,6,1                        ;Load LONLIT micro word
                                00EA  1240          LOADREG         WDREG,2$,,L                     ;Load RDM D Register
                                00F2  1241          ERRLOOP                                         ;Error Loop
                                00F4  1242          FETCH           0                               ;Get the first micro word
                                00F9  1243          BRSTCLK         9                               ;Clock stop location 9
                                00FD  1244          CMPREG          WHREG,10$,J,L                   ;Check the received data
                                0106  1245          IFERROR         4,<<SRM><SRK>>                  ;On error call out SRM and SRK
                                010D  1246          ENDLOOP         I                               ;Continue with loop to increment PLA
                                0110  1247          ENDLOOP         J                               ;Continue with loop to increment SLA
                                0113  1248          SKIP                                            ;End of test
                                011A  1249
                                011A  1250 BEGIN_TEST_DATA
                                011A
                                011A       ;-------------------------------------------------------------------------
                                011A  1251
                                011A  1252 1$:
U 00, 7700,C800,0364,0300,0470,1801  011A  1253 ;% 00    NOP                                             ;NOP Micro Instruction
                                0125  1257 30$:
U 01, 7700,7800,7FFF,FFFF,8470,1802  0125  1258 ;% 01    LONLIT_[0]                                      ;PLATCH data
U 02, 7700,4800,BC47,03D4,0470,1803  0130  1262 ;% 02    ROT/SL.PL_WB,RSRC/LONLIT,ALPCTL/WX_R.Q_M        ;Set PLATCH
                                013B  1266 60$:
U 03, 7700,7800,7FFF,FFFF,8470,1804  013B  1267 ;% 03    LONLIT_[0]                                      ;SLATCH data
U 04, 7700,0800,B447,03D4,0470,1805  0146  1271 ;% 04    ROT/PL.SL_WB,RSRC/LONLIT,ALPCTL/WX_R.Q_M        ;Set SLATCH
U 05, 5700,0840,0364,030C,0470,1806  0151  1275 ;% 05    R[TEMP3]_RDM                                    ;RBUS data
                                015C  1279 70$:
U 06, 7700,7800,7FFF,FFFF,8470,1807  015C  1280 ;% 06    LONLIT_[0]                                      ;MBUS data
U 07, 7700,8868,0247,03D4,0470,1808  0167  1284 ;% 07    M[TEMP8]_LONLIT
U 08, 6700,0803,0770,030C,0470,1809  0172  1288 ;% 08    RDM_WB,ROT/XZ.MM,ALPCTL/WX_S,MSRC/TEMP3,RSRC/TEMP3 ;Test micro word
U 09, 7300,4800,0364,0300,0470,180A  017D  1292 ;% 09    NOP,STOP.CLOCK                                  ;Stop clock
                                0188  1296
                        FFFFFFFF  0188  1297 2$:     .Long   ^xFFFFFFFF              ;RBUS data
                                018C  1298
                        FFFFFFFF  018C  1299 22$:    .Long   ^xFFFFFFFF              ;RBUS data
                                0190  1300
                              01  0190  1301 3$:     .Byte   1                       ;PLATCH data
                                0191  1302
                              00  0191  1303 4$:     .Byte   0                       ;PLATCH data
                                0192  1304
```

```
                08 07 06 05 04 03 02 01  0192  1305 5$:     .Byte   01,02,03,04,05,06,07,08 ;PLATCH data
                10 0F 0E 0D 0C 0B 0A 09  019A  1306         .Byte   09,10,11,12,13,14,15,16 ;PLATCH data
                18 17 16 15 14 13 12 11  01A2  1307         .Byte   17,18,19,20,21,22,23,24 ;PLATCH data
                20 1F 1E 1D 1C 1B 1A 19  01AA  1308         .Byte   25,26,27,28,29,30,31,32 ;PLATCH data
                                         01B2  1309 ;
                                         01B2  1310 ; Expected Data
                                         01B2  1311 ;
00000005 00000001 00000000 00000000      01B2  1312 10$:     .Long   ^x00000000,^x00000000,^x00000001,^x00000005,^x0000000D
                                  0000000D 01C2
000000FD 0000007D 0000003D 0000001D      01C6  1313         .Long   ^x0000001D,^x0000003D,^x0000007D,^x000000FD,^x000001FD
                                  000001FD 01D6
00001FFD 00000FFD 000007FD 000003FD      01DA  1314         .Long   ^x000003FD,^x000007FD,^x00000FFD,^x00001FFD,^x00003FFD
                                  00003FFD 01EA
0003FFFD 0001FFFD 0000FFFD 00007FFD      01EE  1315         .Long   ^x00007FFD,^x0000FFFD,^x0001FFFD,^x0003FFFD,^x0007FFFD
                                  0007FFFD 01FE
007FFFFD 003FFFFD 001FFFFD 000FFFFD      0202  1316         .Long   ^x000FFFFD,^x001FFFFD,^x003FFFFD,^x007FFFFD,^x00FFFFFD
                                  00FFFFFD 0212
0FFFFFFD 07FFFFFD 03FFFFFD 01FFFFFD      0216  1317         .Long   ^x01FFFFFD,^x03FFFFFD,^x07FFFFFD,^x0FFFFFFD,^x1FFFFFFD
                                  1FFFFFFD 0226
         FFFFFFFD 7FFFFFFD 3FFFFFFD      022A  1318         .Long   ^x3FFFFFFD,^x7FFFFFFD,^xFFFFFFFD
                                         0236  1319
                           00000000      0236  1320 20$:     .Long   0                       ;Index to increment PLATCH saved here
                                         023A  1321
                           FFFFFFFF      023A  1322 40$:     .Long   ^xFFFFFFFF              ;Data to Initialize 2$ and 22$
                                         023E  1323
                           00000000      023E  1324 50$:     .Long   0                       ;Index to increment SLATCH saved here
                                         0242  1325 ;
                                         0242  1326 ; loop 32-bit 0 = 0
                                         0242  1327 ;
FF FF FF FF FF FF FF FF FF FF FF FF      0242  1328 99$:     .Byte   ^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF
FF FF FF FF FF FF FF FF FF FF FF FF      024E  1329         .Byte   ^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF
            FE FF FF FF FF FF FF FF      025A  1330         .Byte   ^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFE
                                         0262  1331
                                         0262  1332 BEGIN_CPU_DATA
                                         0002  1333
                                         0002  1334 RTEMP_DATA
                                         0001  1335
                           00000000      0001  1336         .Long   0                       ;RTEMP0
                           00000000      0005  1337         .Long   0                       ;RTEMP1
                           FFFFFFFF      0009  1338         .Long   ^xFFFFFFFF              ;RTEMP2
                           00000000      000D  1339         .Long   0                       ;RTEMP3
                                         0011  1340
                                         0011  1341 END_CPU_DATA
                                         0262  1342
                                         0262  1343 END_TEST_DATA
                                         0262
                                         0262        ;-------------------------------------------------------------------------
                                         0262  1344
                                         0262  1345 ENDTEST
```

```
                        003D        .SBTTL  "TEST   041     Rotator Test 15 Extract RBUS concatenated with RBUS Test
                        003D  1347 ;*******************************************************************************
                        003D  1348 ;++
                        003D  1349 ;
                        003D  1350 ;  FUNCTIONAL DESCRIPTION
                        003D  1351 ;
                        003D  1352 ;       This test checks that the Super Rotator will extract a bit field from
                        003D  1353 ;       RBUS'RBUS. The  STARTING BIT POSITION  is specified by the PLATCH which
                        003D  1354 ;       will be incremented from 0 to 31. The SIZE of the field is specified by
                        003D  1355 ;       the  SLATCH  which will be incremented from 1 to 31. For each increment
                        003D  1356 ;       of the SLATCH the PLATCH will be incremented 32 times thus rotating the
                        003D  1357 ;       STARTING BIT POSITION  across the  RBUS'RBUS from RBUS<00> to RBUS<31>.
                        003D  1358 ;       Incrementing the  SLATCH  will select a field width from 1 to 32  bits.
                        003D  1359 ;
                        003D  1360 ;  TEST ALGORITHM
                        003D  1361 ;
                        003D  1362 ;       1. Load the micro words to perform the test.
                        003D  1363 ;       2. Setup a loop of 29 iterations.
                        003D  1364 ;       3. Select and load values from the tables.
                        003D  1365 ;       4. Setup another loop of 31 iterations.
                        003D  1366 ;       5. Select and load values from the tables.
                        003D  1367 ;       6. Execute the micro sequence.
                        003D  1368 ;               1. Increment SLATCH.
                        003D  1369 ;               2. Increment PLATCH
                        003D  1370 ;               3. Extract and Zero Extend data onto WBUS.
                        003D  1371 ;       7. Check the results of the WBUS read.
                        003D  1372 ;       8. Looped 32 times? If not go to step five.
                        003D  1373 ;       9. Looped 29 times? If not go to step three.
                        003D  1374 ;
                        003D  1375 ;  DATA PATTERNS
                        003D  1376 ;
                        003D  1377 ;       FFFFFFF
                        003D  1378 ;            01
                        003D  1379 ;
                        003D  1380 ;  LOGIC DESCRIPTION
                        003D  1381 ;
                        003D  1382 ;       These subtests check the gates that effect the Extract and Zero Extend
                        003D  1383 ;       function.
                        003D  1384 ;
                        003D  1385 ;  ASSUMPTIONS
                        003D  1386 ;
                        003D  1387 ;       This test requires no signals from any other CPU Module but assumes
                        003D  1388 ;       an operational RBUS, MBUS, SBUS and WBUS.
                        003D  1389 ;
                        003D  1390 ;  ERROR DESCRIPTION
                        003D  1391 ;
                        003D  1392 ;       EXPECTED DATA (See 10$)
                        003D  1393 ;       RECEIVED DATA
                        003D  1394 ;       LOOP COUNT I
                        003D  1395 ;       LOOP COUNT J
                        003D  1396 ;       RTEMPS 0-4
                        003D  1397 ;
                        003D  1398 ;--
                        003D  1399 ;*******************************************************************************
                        003D  1400 ;///////////////////////////////////////////////////////////////////////////////
```

```
                                  003D
                                  003D          INITIALIZE                                  ;DCLO low,Init Scratchpads
                                  003F          LOOP            J,3,32                      ;Set loop count to increment SLATCH
                                  0046          SAVEINDEX       J,50$                       ;Save loop count to increment SLATCH
                                  004B          LDFIELD         40$,,L,2$,00,31,,,,,NS      ;Initialize 2$
                                  005C          LDFIELD         40$,,L,22$,00,31,,,,,NS     ;Initialize 22$
                                  006D          LOADDCS         1$,,0,<^xA>                 ;Load the test micro instructions
                                  0074          LDFIELD         50$,,L,60$,31,62,LL         ;Load LONLIT field
                                  0085          LOADDCS         60$,,3,1                    ;Load LONLIT micro word
                                  008C          LOOP            I,1,32                      ;Set loop count to increment PLATCH
                                  0093          SAVEINDEX       I,20$                       ;Save loop count to increment PLATCH
                                  0098          LDFIELD         20$,,L,30$,31,62,LL         ;Load LONLIT field
                                  00A9          LOADDCS         30$,,1,1                    ;Load LONLIT micro word
                                  00B0          LDFIELD         99$,I,B,2$,0,7,,,,,NS       ;Load LSByte of 99$
                                  00C1          LDFIELD         3$,,B,2$,4$,5$,,I,I,,NS     ;Load PLATCH data
                                  00D2          LDFIELD         22$,,L,70$,31,62,LL         ;Load LONLIT field
                                  00E3          LOADDCS         70$,,6,1                    ;Load MBUS data
                                  00EA          LOADREG         WDREG,2$,,L                 ;Load RBUS data
                                  00F2          ERRLOOP                                     ;Error Loop
                                  00F4          FETCH           0                           ;Get first micro word
                                  00F9          BRSTCLK         9                           ;Clock stop location 9
                                  00FD          CMPREG          WHREG,10$,J,L               ;Check the received data
                                  0106          IFERROR         4,<<SRM><SRK>>              ;On error call out SRM and SRK
                                  010D          ENDLOOP         I                           ;Continue with loop to increment PLA
                                  0110          ENDLOOP         J                           ;Continue with loop to increment SLA
                                  0113          SKIP                                        ;End of test
                                  011A
                                  011A  BEGIN_TEST_DATA
                                  011A
                                  011A      ;-----------------------------------------------------------------------
                                  011A
                                  011A  1$:
U 00, 7700,C800,0364,0300,0470,1801  011A  ;% 00   NOP                                          ;NOP Micro Instruction
                                  0125  30$:
U 01, 7700,7800,7FFF,FFFF,8470,1802  0125  ;% 01   LONLIT_[0]                                   ;PLATCH data
U 02, 7700,4800,BC47,03D4,0470,1803  0130  ;% 02   ROT/SL.PL_WB,RSRC/LONLIT,ALPCTL/WX_R.Q_M     ;Set PLATCH
                                  013B  60$:
U 03, 7700,7800,7FFF,FFFF,8470,1804  013B  ;% 03   LONLIT_[0]                                   ;SLATCH data
U 04, 7700,0800,B447,03D4,0470,1805  0146  ;% 04   ROT/PL.SL_WB,RSRC/LONLIT,ALPCTL/WX_R.Q_M     ;Set SLATCH
U 05, 5700,0840,0364,030C,0470,1806  0151  ;% 05   R[TEMP3]_RDM                                 ;RBUS data
                                  015C  70$:
U 06, 7700,7800,7FFF,FFFF,8470,1807  015C  ;% 06   LONLIT_[0]                                   ;MBUS data
U 07, 7700,8868,0247,03D4,0470,1808  0167  ;% 07   M[TEMP8]_LONLIT
U 08, 6700,4808,0B70,030C,0470,1809  0172  ;% 08   RDM_WB,ROT/XZ.RR,ALPCTL/WX_S,MSRC/TEMP8,RSRC/TEMP3 ;Test micro word
U 09, 7300,4800,0364,0300,0470,180A  017D  ;% 09   NOP,STOP.CLOCK                              ;Stop clock
                                  0188
                         FFFFFFFF  0188  2$:     .Long   ^xFFFFFFFF              ;RBUS data
                                  018C
                         FFFFFFFF  018C  22$:    .Long   ^xFFFFFFFF              ;RBUS data
                                  0190
                               01  0190  3$:     .Byte   1                       ;PLATCH data
                                  0191
                               00  0191  4$:     .Byte   0                       ;PLATCH data
                                  0192
          08 07 06 05 04 03 02 01  0192  5$:     .Byte   01,02,03,04,05,06,07,08 ;PLATCH data
```

```
          10 0F 0E 0D 0C 0B 0A 09  019A  1484          .Byte   09,10,11,12,13,14,15,16 ;PLATCH data
          18 17 16 15 14 13 12 11  01A2  1485          .Byte   17,18,19,20,21,22,23,24 ;PLATCH data
          20 1F 1E 1D 1C 1B 1A 19  01AA  1486          .Byte   25,26,27,28,29,30,31,32 ;PLATCH data
                                   01B2  1487 ;
                                   01B2  1488 ; Expected Data
                                   01B2  1489 ;
00000005 00000001 00000000 00000000  01B2  1490 10$:     .Long   ^x00000000,^x00000000,^x00000001,^x00000005,^x0000000D
                           0000000D  01C2
000000FD 0000007D 0000003D 0000001D  01C6  1491          .Long   ^x0000001D,^x0000003D,^x0000007D,^x000000FD,^x000001FD
                           000001FD  01D6
00001FFD 00000FFD 000007FD 000003FD  01DA  1492          .Long   ^x000003FD,^x000007FD,^x00000FFD,^x00001FFD,^x00003FFD
                           00003FFD  01EA
0003FFFD 0001FFFD 0000FFFD 00007FFD  01EE  1493          .Long   ^x00007FFD,^x0000FFFD,^x0001FFFD,^x0003FFFD,^x0007FFFD
                           0007FFFD  01FE
007FFFFD 003FFFFD 001FFFFD 000FFFFD  0202  1494          .Long   ^x000FFFFD,^x001FFFFD,^x003FFFFD,^x007FFFFD,^x00FFFFFD
                           00FFFFFD  0212
0FFFFFFD 07FFFFFD 03FFFFFD 01FFFFFD  0216  1495          .Long   ^x01FFFFFD,^x03FFFFFD,^x07FFFFFD,^x0FFFFFFD,^x1FFFFFFD
                           1FFFFFFD  0226
         FFFFFFFD 7FFFFFFD 3FFFFFFD  022A  1496          .Long   ^x3FFFFFFD,^x7FFFFFFD,^xFFFFFFFD
                                   0236  1497
                           00000000  0236  1498 20$:     .Long   0                       ;Index to increment PLATCH saved here
                                   023A  1499
                           FFFFFFFF  023A  1500 40$:     .Long   ^xFFFFFFFF              ;Data to initialize 2$ and 22$
                                   023E  1501
                           00000000  023E  1502 50$:     .Long   0                       ;Index to increment PLATCH saved here
                                   0242  1503 ;
                                   0242  1504 ; loop 32 - bit 0 = 0
                                   0242  1505 ;
FF FF FF FF FF FF FF FF FF FF FF FF  0242  1506 99$:     .Byte   ^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF
FF FF FF FF FF FF FF FF FF FF FF FF  024E  1507          .Byte   ^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF
            FE FF FF FF FF FF FF FF  025A  1508          .Byte   ^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFF,^xFE
                                   0262  1509
                                   0262  1510 BEGIN_CPU_DATA
                                   0002  1511
                                   0002  1512 RTEMP_DATA
                                   0001  1513
                           00000000  0001  1514          .Long   0                       ;RTEMP0
                           00000000  0005  1515          .Long   0                       ;RTEMP1
                           FFFFFFFF  0009  1516          .Long   ^xFFFFFFFF              ;RTEMP2
                           00000000  000D  1517          .Long   0                       ;RTEMP3
                                   0011  1518
                                   0011  1519 END_CPU_DATA
                                   0262  1520
                                   0262  1521 END_TEST_DATA
                                   0262
                                   0262        ;-------------------------------------------------------------------------
                                   0262  1522
                                   0262  1523 ENDTEST
```

```
                        002D        .SBTTL  "TEST   042     Rotator Test 16 Extract PTE Address Test
                        002D  1525 ;*******************************************************************************
                        002D  1526 ;++
                        002D  1527 ;
                        002D  1528 ; FUNCTIONAL DESCRIPTION
                        002D  1529 ;
                        002D  1530 ;       This test examines the rotator function used to extract PTE addresses
                        002D  1531 ;       for the memory mapping routines. A 23 bit field is extracted from
                        002D  1532 ;       MBUS <29:07> and is driven out on SBUS <22:00>. The remainder of the
                        002D  1533 ;       SBUS is zero extended.
                        002D  1534 ;
                        002D  1535 ; TEST ALGORITHM
                        002D  1536 ;
                        002D  1537 ;       1. Setup a loop of 6 iterations.
                        002D  1538 ;       2. Generate a stuck at zero or one test pattern.
                        002D  1539 ;       3. Load the test pattern into the RDM's WDREG.
                        002D  1540 ;       4. Execute the following micro-sequence:
                        002D  1541 ;               1. Write the test pattern into MTEMP0.
                        002D  1542 ;               2. Run the test pattern through the Rotator and read it back
                        002D  1543 ;                  into the RDM.
                        002D  1544 ;       5. Check the result.
                        002D  1545 ;       6. Loop until done.
                        002D  1546 ;
                        002D  1547 ; TEST DATA
                        002D  1548 ;
                        002D  1549 ;       Because SBUS<22:00> <- MBUS <29:07> and SBUS<31:23> <- 0 the results
                        002D  1550 ;       read back will be different from the standard test patterns read in.
                        002D  1551 ;
                        002D  1552 ;       Below is a table which shows what the expected data will be:
                        002D  1553 ;
                        002D  1554 ;           MBUS Test Pattern          SBUS Results
                        002D  1555 ;
                        002D  1556 ;               AAAAAAAA                   00555555
                        002D  1557 ;               55555555                   002AAAAA
                        002D  1558 ;               33333333                   00666666
                        002D  1559 ;               0F0F0F0F                   001E1E1E
                        002D  1560 ;               00FF00FF                   0001FE01
                        002D  1561 ;               0000FFFF                   000001FF
                        002D  1562 ;
                        002D  1563 ; LOGICAL DESCRIPTION
                        002D  1564 ;
                        002D  1565 ;       The logical involved is internal to the SRM gate array. SRM in turn
                        002D  1566 ;       gets some of its control signals from the SRK gate array.
                        002D  1567 ;
                        002D  1568 ; ASSUMPTIONS
                        002D  1569 ;
                        002D  1570 ;       This test requires an operational MBUS, SBUS, WBUS and the ALPCTL
                        002D  1571 ;       special function WMUX <- BMUX.
                        002D  1572 ;
                        002D  1573 ; ERROR DESCRIPTION
                        002D  1574 ;
                        002D  1575 ;       On error this test will call out the SRM and SRK gate arrays.
                        002D  1576 ;       The following information will also by typed out:
                        002D  1577 ;
                        002D  1578 ;       EXPECTED SBUS DATA
```

```
                                002D  ;       RECEIVED SBUS DATA
                                002D  ;       LOOP COUNT
                                002D  ;--
                                002D  ;*******************************************************************************
                                002D  ;///////////////////////////////////////////////////////////////////////////////
                                002D
                                002D          INITIALIZE                                  ; INITIALIZE THE CPU
                                002F          LOOP            I,1,6                       ; LOOP FOR 6 SA01 TEST PATTERNS
                                0036          SA01PAT         I,1$                        ; GENERATE A SA01 TEST PATTERN
                                003E          LOADREG         WDREG,1$,,L                 ; LOAD THE PATTERN INTO WDREG
                                0046          ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                                0048          FETCH           0                           ; RUN THE PATTERN THROUGH THE
                                004D          BRSTCLK         3                           ; ROTATOR AND INTO WHREG
                                0051          CMPREG          WHREG,2$,I,L                ; IS THE SBUS DATA VALID?
                                005A          IFERROR         ,<<SRM><SRK>>               ; IF NOT, SIGNAL AN ERROR
                                0061          ENDLOOP         I                           ; CONTINUE LOOP
                                0064          SKIP                                        ; END OF TEST
                                006B
                                006B  BEGIN_TEST_DATA
                                006B
                                006B      ;-----------------------------------------------------------------------
                                006B
                        00000000  006B  1$:     .LONG   0       ; LOCATION TO HOLD SA01 PATTERN
                                006F
                                006F  ; Expected SBUS Data
                                006F
                        00555555  006F  2$:     .LONG   ^X00555555
                        002AAAAA  0073          .LONG   ^X002AAAAA
                        00666666  0077          .LONG   ^X00666666
                        001E1E1E  007B          .LONG   ^X001E1E1E
                        0001FE01  007F          .LONG   ^X0001FE01
                        000001FF  0083          .LONG   ^X000001FF
                                0087
                                0087  BEGIN_CPU_DATA
                                0002
                                0002  DCS_DATA
                                0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP                                     ; GET CPU UP TO SPEED
U 01, 5700,C860,0364,0300,0470,1802  000C  ;% 01   M[TEMP0]_RDM                           ; MTEMP0 = TEST PATTERN
U 02, 6700,0800,3347,0300,0470,1803  0017  ;% 02   MSRC/TEMP0,ROT/XZ.PTX,ALPCTL/WX_S.Q_0,RDM_WB  ; WHREG = SBUS DATA
U 03, 7300,C800,0364,0300,0470,1804  0022  ;% 03   STOP.CLOCK                             ; HALT
                                002D
                                002D  RTEMP_DATA
                                0001
                        00000000  0001          .LONG   0       ; INITIAL VALUE FOR MTEMP0
                                0005
                                0005  END_CPU_DATA
                                0087
                                0087  END_TEST_DATA
                                0087
                                0087      ;-----------------------------------------------------------------------
                                0087
                                0087  ENDTEST
```

```
0035          .SBTTL  "TEST   043     Rotator Test 17 Extract Virtual Page Number Test
0035  1641 ;*****************************************************************************
0035  1642 ;++
0035  1643 ;
0035  1644 ; FUNCTIONAL DESCRIPTION
0035  1645 ;
0035  1646 ;       This test examines the rotator function used to extract the VPN field
0035  1647 ;       from a virtual address. A 21 bit field is extracted from MBUS <29:09>
0035  1648 ;       and is driven out on SBUS <20:00>. The remainder of the SBUS is zero
0035  1649 ;       extended.
0035  1650 ;
0035  1651 ; TEST ALGORITHM
0035  1652 ;
0035  1653 ;       1. Setup a loop of six iterations.
0035  1654 ;       2. Generate a stuck at zero or one test pattern.
0035  1655 ;       3. Load the test pattern into the RDM's WDREG.
0035  1656 ;       4. Execute the following micro-sequence:
0035  1657 ;               1. Load the test pattern from WDREG into MTEMP0.
0035  1658 ;               2. Execute the VPN extract on MTEMP0 and read the result back
0035  1659 ;                  into the RDM's WHREG.
0035  1660 ;       5. Check the result and signal an error if wrong.
0035  1661 ;       6. Loop back until done.
0035  1662 ;
0035  1663 ; TEST DATA
0035  1664 ;
0035  1665 ;       Because SBUS<20:00> <- MBUS <29:09> and SBUS<31:21> <- 0 the results
0035  1666 ;       read back will be different from the standard test patterns read in.
0035  1667 ;
0035  1668 ;       Below is a table which shows what the expected data will be:
0035  1669 ;
0035  1670 ;           MBUS Test Pattern      Expected SBUS Results
0035  1671 ;
0035  1672 ;                AAAAAAAA                00155555
0035  1673 ;                55555555                000AAAAA
0035  1674 ;                33333333                00199999
0035  1675 ;                0F0F0F0F                00078787
0035  1676 ;                00FF00FF                00007F80
0035  1677 ;                0000FFFF                0000007F
0035  1678 ;
0035  1679 ; LOGICAL DESCRIPTION
0035  1680 ;
0035  1681 ;       The logic involved is internal to the SRM gate array. SRM in turn
0035  1682 ;       gets some of its control signals from the SRK gate array.
0035  1683 ;
0035  1684 ; ASSUMPTIONS
0035  1685 ;
0035  1686 ;       This test requires an operational MBUS, SBUS, WBUS, the ALPCTL special
0035  1687 ;       function WMUX <- BMUX and the MTEMP0 scratchpad.
0035  1688 ;
0035  1689 ; ERROR DESCRIPTION
0035  1690 ;
0035  1691 ;       On error this test will call out the SRM and SRK gate arrays.
0035  1692 ;       The following data will also be typed out:
0035  1693 ;
0035  1694 ;       EXPECTED SBUS DATA
```

```
                                  0035  ;       RECEIVED SBUS DATA
                                  0035  ;       LOOP COUNT
                                  0035  ;
                                  0035  ;--
                                  0035  ;*******************************************************************************
                                  0035  ;///////////////////////////////////////////////////////////////////////////////
                                  0035
                                  0035          INITIALIZE                                  ; INITIALIZE THE CPU
                                  0037          LOOP            I,1,6                       ; LOOP FOR 6 SA01 PATTERNS
                                  003E          SA01PAT         I,1$                        ; GENERATE A SA01 PATTERN
                                  0046          LOADREG         WDREG,1$,,L                 ; LOAD THE PATTERN INTO WDREG
                                  004E          ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                                  0050          FETCH           0                           ; EXECUTE THE VPN EXTRACT AND
                                  0055          BRSTCLK         3                           ; READ THE RESULT BACK TO WHREG
                                  0059          CMPREG          WHREG,2$,I,L                ; IS THE RESULT CORRECT?
                                  0062          IFERROR         ,<<SRM><SRK>>               ; IF NOT, SIGNAL ERROR
                                  0069          ENDLOOP         I                           ; CONTINUE LOOP
                                  006C          SKIP                                        ; END OF TEST
                                  0073
                                  0073  BEGIN_TEST_DATA
                                  0073
                                  0073          ;-----------------------------------------------------------------------
                                  0073
                        00000000  0073  1$:     .LONG   0       ; LOCATION TO HOLD SA01 PATTERN
                                  0077
                                  0077  ; Expected SBUS data
                                  0077
                        00155555  0077  2$:     .LONG   ^X00155555
                        000AAAAA  007B          .LONG   ^X000AAAAA
                        00199999  007F          .LONG   ^X00199999
                        00078787  0083          .LONG   ^X00078787
                        00007F80  0087          .LONG   ^X00007F80
                        0000007F  008B          .LONG   ^X0000007F
                                  008F
                                  008F  BEGIN_CPU_DATA
                                  0002
                                  0002  DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP                                     ; GET CPU UP TO SPEED
U 01, 5700,C860,0364,0300,0470,1802  000C  ;% 01   M[TEMP0]_RDM                            ; MTEMP0 = TEST PATTERN
U 02, 6700,4800,3747,0300,0470,1803  0017  ;% 02   MSRC/TEMP0,ROT/XZ.VPN,ALPCTL/WX_S.Q_0,RDM_WB  ; WHREG = SBUS RESULT
U 03, 7300,C800,0364,0300,0470,1804  0022  ;% 03   STOP.CLOCK                              ; HALT
                                  002D
                                  002D  RTEMP_DATA
                                  0001
                        00000000  0001          .LONG   0       ; INITIAL VALUE FOR MTEMP0
                                  0005
                                  0005  END_CPU_DATA
                                  008F
                                  008F  END_TEST_DATA
                                  008F
                                  008F          ;-----------------------------------------------------------------------
                                  008F
                                  008F  ENDTEST
```

```
0028          .SBTTL  "TEST   044     Rotator Test 18 Nibble Extract Test
0028  1758 ;*******************************************************************************
0028  1759 ;++
0028  1760 ;
0028  1761 ; FUNCTIONAL DESCRIPTION
0028  1762 ;
0028  1763 ;       This test examines the rotator function used to extract and zero extend
0028  1764 ;       the low order nibble from the MBUS.
0028  1765 ;
0028  1766 ; TEST ALGORITHM
0028  1767 ;
0028  1768 ;       1. Setup a loop of six iterations.
0028  1769 ;       2. Generate a stuck at zero or one test pattern.
0028  1770 ;       3. Load the test pattern into the RDM's WDREG.
0028  1771 ;       4. Execute the following micro-sequence:
0028  1772 ;               1. Load the test pattern from WDREG into MTEMP0.
0028  1773 ;               2. Execute the nibble extract on MTEMP0 and read the result
0028  1774 ;                  back into the RDM's WHREG.
0028  1775 ;       5. Check the result and signal an error if wrong.
0028  1776 ;       6. Loop back until done.
0028  1777 ;
0028  1778 ; TEST DATA
0028  1779 ;
0028  1780 ;       Below is a table which shows what the expected data will be:
0028  1781 ;
0028  1782 ;           MBUS Test Pattern      Expected SBUS Results
0028  1783 ;
0028  1784 ;               AAAAAAAA                0000000A
0028  1785 ;               55555555                00000005
0028  1786 ;               33333333                00000003
0028  1787 ;               0F0F0F0F                0000000F
0028  1788 ;               00FF00FF                0000000F
0028  1789 ;               0000FFFF                0000000F
0028  1790 ;
0028  1791 ; LOGICAL DESCRIPTION
0028  1792 ;
0028  1793 ;       The logic involved is internal to the SRM gate array. SRM in turn
0028  1794 ;       gets some of its control signals from the SRK gate array.
0028  1795 ;
0028  1796 ; ASSUMPTIONS
0028  1797 ;
0028  1798 ;       This test requires an operational MBUS, SBUS, WBUS, the ALPCTL special
0028  1799 ;       function WMUX <- BMUX and the MTEMP0 scratchpad.
0028  1800 ;
0028  1801 ; ERROR DESCRIPTION
0028  1802 ;
0028  1803 ;       On error this test will call out the SRM and SRK gate arrays.
0028  1804 ;       The following data will also be typed out:
0028  1805 ;
0028  1806 ;       EXPECTED SBUS DATA
0028  1807 ;       RECEIVED SBUS DATA
0028  1808 ;       LOOP COUNT
0028  1809 ;
0028  1810 ;--
0028  1811 ;*******************************************************************************
```

```
                        0028  1812 ;////////////////////////////////////////////////////////////////////////////////
                        0028  1813
                        0028  1814         INITIALIZE                                      ; INITIALIZE THE CPU
                        002A  1815         LOOP            I,1,6                           ; LOOP FOR 6 SA01 PATTERNS
                        0031  1816         SA01PAT         I,1$                            ; GENERATE A SA01 PATTERN
                        0039  1817         LOADREG         WDREG,1$,,L                     ; LOAD THE PATTERN INTO WDREG
                        0041  1818         ERRLOOP                                         ; LOOP ON ERROR FROM HERE
                        0043  1819         FETCH           0                               ; EXECUTE THE NIBBLE EXTRACT &
                        0048  1820         BRSTCLK         3                               ; READ THE RESULT INTO WHREG
                        004C  1821         CMPREG          WHREG,2$,I,L                    ; IS THE SBUS DATA CORRECT?
                        0055  1822         IFERROR         ,<<SRM><SRK>>                   ; IF NOT, SIGNAL AN ERROR
                        005C  1823         ENDLOOP         I                               ; CONTINUE LOOP
                        005F  1824         SKIP                                            ; END OF TEST
                        0066  1825
                        0066  1826 BEGIN_TEST_DATA
                        0066
                        0066       ;-------------------------------------------------------------------------
                        0066  1827
                00000000  0066  1828 1$:      .LONG   0       ; LOCATION TO HOLD SA01 PATTERN
                        006A  1829
                        006A  1830 ; Expected SBUS data
                        006A  1831
                0000000A  006A  1832 2$:      .LONG   ^X0000000A
                00000005  006E  1833          .LONG   ^X00000005
                00000003  0072  1834          .LONG   ^X00000003
                0000000F  0076  1835          .LONG   ^X0000000F
                0000000F  007A  1836          .LONG   ^X0000000F
                0000000F  007E  1837          .LONG   ^X0000000F
                        0082  1838
                        0082  1839 BEGIN_CPU_DATA
                        0002  1840
                        0002  1841 DCS_DATA
                        0001  1842
U 00, 7700,C800,0364,0300,0470,1801     0001  1843 ;% 00   NOP                                             ; GET CPU UP TO SPEED
U 01, 5700,C860,0364,0300,0470,1802     000C  1847 ;% 01   M[TEMP0]_RDM                                   ; MTEMP0 = TEST PATTERN
U 02, 6700,0800,3F47,0300,0470,1803     0017  1851 ;% 02   MSRC/TEMP0,ROT/GETNIB,ALPCTL/WX_S.Q_0,RDM_WB  ; WHREG = SBUS RESULT
U 03, 7300,C800,0364,0300,0470,1804     0022  1855 ;% 03   STOP.CLOCK                                    ; HALT
                        002D  1859
                        002D  1860 RTEMP_DATA
                        0001  1861
                00000000  0001  1862          .LONG   0       ; INITIAL VALUE FOR MTEMP0
                        0005  1863
                        0005  1864 END_CPU_DATA
                        0082  1865
                        0082  1866 END_TEST_DATA
                        0082
                        0082       ;-------------------------------------------------------------------------
                        0082  1867
                        0082  1868 ENDTEST
```

```
                        002A        .SBTTL  "TEST   045     Rotator Test 19 Exponent Extract Test
                        002A  1870 ;*******************************************************************************
                        002A  1871 ;++
                        002A  1872 ;
                        002A  1873 ; FUNCTIONAL DESCRIPTION
                        002A  1874 ;
                        002A  1875 ;       This test examines the rotator function used to extract and zero extend
                        002A  1876 ;       the exponent field of a floating point number.
                        002A  1877 ;
                        002A  1878 ; TEST ALGORITHM
                        002A  1879 ;
                        002A  1880 ;       1. Setup a loop of six iterations.
                        002A  1881 ;       2. Generate a stuck at zero or one test pattern.
                        002A  1882 ;       3. Load the test pattern into the RDM's WDREG.
                        002A  1883 ;       4. Execute the following micro-sequence:
                        002A  1884 ;               1. Load the test pattern from WDREG into MTEMP0.
                        002A  1885 ;               2. Execute the exponent extract on MTEMP0 and read the result
                        002A  1886 ;                  back into the RDM's WHREG.
                        002A  1887 ;       5. Check the result and signal an error if wrong.
                        002A  1888 ;       6. Loop back until done.
                        002A  1889 ;
                        002A  1890 ; TEST DATA
                        002A  1891 ;
                        002A  1892 ;       The rotator function GETEXP works by taking an eight bit field from
                        002A  1893 ;       MBUS <14:7> and then zero extends it and places the result on SBUS
                        002A  1894 ;       <7:0> with SUBS <31:08> equal to 0. This will cause the results read
                        002A  1895 ;       back to be different from the standard test patterns read in.
                        002A  1896 ;
                        002A  1897 ;       Below is a table which shows what the received data should be:
                        002A  1898 ;
                        002A  1899 ;           MBUS Test Pattern      Expected SBUS Results
                        002A  1900 ;
                        002A  1901 ;               AAAAAAAA                00000055
                        002A  1902 ;               55555555                000000AA
                        002A  1903 ;               33333333                00000066
                        002A  1904 ;               0F0F0F0F                0000001E
                        002A  1905 ;               00FF00FF                00000001
                        002A  1906 ;               0000FFFF                000000FF
                        002A  1907 ;
                        002A  1908 ; LOGICAL DESCRIPTION
                        002A  1909 ;
                        002A  1910 ;       The logic involved is internal to the SRM gate array. SRM in turn
                        002A  1911 ;       gets some of its control signals from the SRK gate array.
                        002A  1912 ;
                        002A  1913 ; ASSUMPTIONS
                        002A  1914 ;
                        002A  1915 ;       This test requires an operational MBUS, SBUS, WBUS, the ALPCTL special
                        002A  1916 ;       function WMUX <- BMUX and the MTEMP0 scratchpad.
                        002A  1917 ;
                        002A  1918 ; ERROR DESCRIPTION
                        002A  1919 ;
                        002A  1920 ;       On error this test will call out the SRM and SRK gate arrays.
                        002A  1921 ;       The following data will also be typed out:
                        002A  1922 ;
                        002A  1923 ;       EXPECTED SBUS DATA
```

```
                                002A  ;       RECEIVED SBUS DATA
                                002A  ;       LOOP COUNT
                                002A  ;
                                002A  ;--
                                002A  ;*******************************************************************************
                                002A  ;///////////////////////////////////////////////////////////////////////////////
                                002A
                                002A          INITIALIZE                                  ; INITIALIZE THE CPU
                                002C          LOOP            I,1,6                       ; LOOP FOR 6 SA01 PATTERNS
                                0033          SA01PAT         I,1$                        ; GENERATE A SA01 PATTERN
                                003B          LOADREG         WDREG,1$,,L                 ; LOAD TEST PATTERN INTO WDREG
                                0043          ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                                0045          FETCH           0                           ; EXECUTE THE EXPONENT EXTRACT
                                004A          BRSTCLK         3                           ; AND READ RESULT INTO WHREG
                                004E          CMPREG          WHREG,2$,I,L                ; IS THE RESULT CORRECT?
                                0057          IFERROR         ,<<SRM><SRK>>               ; IF NOT, SIGNAL AN ERROR
                                005E          ENDLOOP         I                           ; CONTINUE LOOP
                                0061          SKIP                                        ; END OF TEST
                                0068
                                0068  BEGIN_TEST_DATA
                                0068
                                0068      ;-----------------------------------------------------------------------
                                0068
                     00000000   0068  1$:      .LONG   0       ; LOCATION TO HOLD SA01 PATTERN
                                006C
                                006C  ; Expected SBUS data
                                006C
                     00000055   006C  2$:      .LONG   ^X00000055
                     000000AA   0070          .LONG   ^X000000AA
                     00000066   0074          .LONG   ^X00000066
                     0000001E   0078          .LONG   ^X0000001E
                     00000001   007C          .LONG   ^X00000001
                     000000FF   0080          .LONG   ^X000000FF
                                0084
                                0084  BEGIN_CPU_DATA
                                0002
                                0002  DCS_DATA
                                0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP                                   ; GET CPU UP TO SPEED
U 01, 5700,C860,0364,0300,0470,1802  000C  ;% 01   M[TEMP0]_RDM                          ; MTEMP0 = TEST PATTERN
U 02, 6700,4800,4347,0300,0470,1803  0017  ;% 02   MSRC/TEMP0,ROT/GETEXP,ALPCTL/WX_S.Q_0,RDM_WB  ; WHREG = SBUS
U 03, 7300,C800,0364,0300,0470,1804  0022  ;% 03   STOP.CLOCK                            ; HALT
                                002D
                                002D  RTEMP_DATA
                                0001
                     00000000   0001          .LONG   0       ; INITIAL VALUE FOR MTEMP0
                                0005
                                0005  END_CPU_DATA
                                0084
                                0084  END_TEST_DATA
                                0084
                                0084      ;-----------------------------------------------------------------------
                                0084
                                0084  ENDTEST
```

```
                002A        .SBTTL  "TEST   046     Rotator Test 20 Clear MBUS Bytes Test
                002A  1987 ;*******************************************************************************
                002A  1988 ;++
                002A  1989 ;
                002A  1990 ; FUNCTIONAL DESCRIPTION
                002A  1991 ;
                002A  1992 ;       This test examines the rotator functions that clear bytes of the MBUS.
                002A  1993 ;
                002A  1994 ; TEST ALGORITHM
                002A  1995 ;
                002A  1996 ;       This test has been divided into three subtests. Subtest 1 examines the
                002A  1997 ;       CLR1BM micro-order, Subtest 2 examines the CLR2BM micro-order and
                002A  1998 ;       Subtest 3 examines the CLR3BM micro-order. The algorithm used is the
                002A  1999 ;       same for every subtest.
                002A  2000 ;
                002A  2001 ;       1. Load the rotator micro-instruction for the current subtest into DCS.
                002A  2002 ;       2. Setup a loop for six test patterns.
                002A  2003 ;       3. Generate a stuck at zero or one test pattern.
                002A  2004 ;       4. Load the test pattern into the RDM's WDREG.
                002A  2005 ;       5. Execute the following micro-sequence:
                002A  2006 ;               1. Write the test pattern from WDREG to MTEMP0.
                002A  2007 ;               2. Execute the clear byte function and read the result into the
                002A  2008 ;                  RDM's WHREG.
                002A  2009 ;       6. Check the result and signal an error if wrong.
                002A  2010 ;       7. Loop back until done.
                002A  2011 ;
                002A  2012 ; TEST DATA
                002A  2013 ;
                002A  2014 ;       The following table gives the exepected data for each subtest.
                002A  2015 ;
                002A  2016 ;             MBUS  Data      Subtest  1      Subtest  2      Subtest  3
                002A  2017 ;
                002A  2018 ;              AAAAAAAA        AAAAAA00        AAAA0000        AA000000
                002A  2019 ;              55555555        55555500        55550000        55000000
                002A  2020 ;              33333333        33333300        33330000        33000000
                002A  2021 ;              0F0F0F0F        0F0F0F00        0F0F0000        0F000000
                002A  2022 ;              00FF00FF        00FF0000        00FF0000        00000000
                002A  2023 ;              0000FFFF        0000FF00        00000000        00000000
                002A  2024 ;
                002A  2025 ; LOGICAL DESCRIPTION
                002A  2026 ;
                002A  2027 ;       The logic involved is internal to the SRM gate array. SRM in turn
                002A  2028 ;       gets some of its control signals from the SRK gate array.
                002A  2029 ;
                002A  2030 ; ASSUMPTIONS
                002A  2031 ;
                002A  2032 ;       This test requires an operational MBUS, SBUS, WBUS, the ALPCTL special
                002A  2033 ;       function WMUX <- BMUX and the MTEMP0 scratchpad.
                002A  2034 ;
                002A  2035 ; ERROR DESCRIPTION
                002A  2036 ;
                002A  2037 ;       On error this test will call out the SRM and SRK gate arrays.
                002A  2038 ;       The following data will also be typed out:
                002A  2039 ;
                002A  2040 ;       EXPECTED SBUS DATA
```

```
                        002A  2041 ;       RECEIVED SBUS DATA
                        002A  2042 ;       LOOP COUNT
                        002A  2043 ;
                        002A  2044 ;--
                        002A  2045 ;*******************************************************************************
                        002A  2046
                        002A  2047         SUBTEST         1
                        002D
                        002D       ;////////////////////////////////////////////////////////////////////////////
                        002D  2048 ;+
                        002D  2049 ; This subtest examines the ROT/CLR1BM micro-order.
                        002D  2050 ;-
                        002D  2051         INITIALIZE                              ; INITIALIZE THE CPU
                        002F  2052         LOADDCS         2$,,2,1                 ; LOAD THE TEST MICRO-WORD
                        0036  2053         LOOP            I,1,6                   ; LOOP FOR 6 SA01 PATTERNS
                        003D  2054         SA01PAT         I,1$                    ; GENERATE A SA01 PATTERN
                        0045  2055         LOADREG         WDREG,1$,,L             ; LOAD TEST PATTERN INTO WDREG
                        004D  2056         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        004F  2057         FETCH           0                       ; EXECUTE THE CLEAR MBUS BYTE
                        0054  2058         BRSTCLK         3                       ; AND READ RESULT INTO WHREG
                        0058  2059         CMPREG          WHREG,5$,I,L            ; IS THE RESULT CORRECT?
                        0061  2060         IFERROR         ,<<SRM><SRK>>           ; IF NOT, SIGNAL AN ERROR
                        0068  2061         ENDLOOP         I                       ; CONTINUE LOOP
                        006B  2062
                        006B  2063         SUBTEST         2
                        006E
                        006E       ;////////////////////////////////////////////////////////////////////////////
                        006E  2064 ;+
                        006E  2065 ; This subtest examines the ROT/CLR2BM micro-order.
                        006E  2066 ;-
                        006E  2067         LOADDCS         3$,,2,1                 ; LOAD THE TEST MICRO-WORD
                        0075  2068         LOOP            I,1,6                   ; LOOP FOR 6 SA01 PATTERNS
                        007C  2069         SA01PAT         I,1$                    ; GENERATE A SA01 PATTERN
                        0084  2070         LOADREG         WDREG,1$,,L             ; LOAD TEST PATTERN INTO WHREG
                        008C  2071         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        008E  2072         FETCH           0                       ; EXECUTE THE CLR2BM FUNCTION
                        0093  2073         BRSTCLK         3                       ; AND READ RESULT INTO WHREG
                        0097  2074         CMPREG          WHREG,6$,I,L            ; IS THE RESULT CORRECT?
                        00A0  2075         IFERROR         ,<<SRM><SRK>>           ; IF NOT, SIGNAL ERROR
                        00A7  2076         ENDLOOP         I                       ; CONTINUE LOOP
                        00AA  2077
                        00AA  2078         SUBTEST         3
                        00AD
                        00AD       ;////////////////////////////////////////////////////////////////////////////
                        00AD  2079 ;+
                        00AD  2080 ; This subtest examines the ROT/CLR3BM micro-order.
                        00AD  2081 ;-
                        00AD  2082         LOADDCS         4$,,2,1                 ; LOAD THE TEST MICRO-WORD
                        00B4  2083         LOOP            I,1,6                   ; LOOP FOR 6 SA01 PATTERNS
                        00BB  2084         SA01PAT         I,1$                    ; GENERATE A SA01 PATTERN
                        00C3  2085         LOADREG         WDREG,1$,,L             ; LOAD TEST PATTERN INTO WDREG
                        00CB  2086         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        00CD  2087         FETCH           0                       ; EXECUTE THE CLR3BM FUNCTION
                        00D2  2088         BRSTCLK         3                       ; AND READ RESULT INTO WHREG
                        00D6  2089         CMPREG          WHREG,7$,I,L            ; IS THE RESULT CORRECT?
```

```
                                00DF  2090          IFERROR         ,<<SRM><SRK>>           ; IF NOT, SIGNAL AN ERROR
                                00E6  2091          ENDLOOP         I                       ; CONTINUE LOOP
                                00E9  2092          SKIP                                    ; END OF TEST
                                00F0  2093
                                00F0  2094 BEGIN_TEST_DATA
                                00F0
                                00F0          ;-------------------------------------------------------------------------
                                00F0  2095
                     00000000   00F0  2096 1$:      .LONG   0       ; LOCATION TO HOLD SA01 PATTERN
                                00F4  2097
                                00F4  2098 2$:
U 02, 6700,4800,4F47,0300,0470,1803 00F4  2099 ;% 02   MSRC/TEMP0,ROT/CLR1BM,ALPCTL/WX_S.Q_0,RDM_WB    ; SUBTEST 1
                                00FF  2103 3$:
U 02, 6700,0800,4B47,0300,0470,1803 00FF  2104 ;% 02   MSRC/TEMP0,ROT/CLR2BM,ALPCTL/WX_S.Q_0,RDM_WB    ; SUBTEST 2
                                010A  2108 4$:
U 02, 6700,0800,5347,0300,0470,1803 010A  2109 ;% 02   MSRC/TEMP0,ROT/CLR3BM,ALPCTL/WX_S.Q_0,RDM_WB    ; SUBTEST 3
                                0115  2113
                                0115  2114 ; Expected data for Subtest 1
                                0115  2115
                     AAAAAA00   0115  2116 5$:      .LONG   ^XAAAAAA00
                     55555500   0119  2117          .LONG   ^X55555500
                     33333300   011D  2118          .LONG   ^X33333300
                     0F0F0F00   0121  2119          .LONG   ^X0F0F0F00
                     00FF0000   0125  2120          .LONG   ^X00FF0000
                     0000FF00   0129  2121          .LONG   ^X0000FF00
                                012D  2122
                                012D  2123 ; Expected data for Subtest 2
                                012D  2124
                     AAAA0000   012D  2125 6$:      .LONG   ^XAAAA0000
                     55550000   0131  2126          .LONG   ^X55550000
                     33330000   0135  2127          .LONG   ^X33330000
                     0F0F0000   0139  2128          .LONG   ^X0F0F0000
                     00FF0000   013D  2129          .LONG   ^X00FF0000
                     00000000   0141  2130          .LONG   ^X00000000
                                0145  2131
                                0145  2132 ; Expected data for Subtest 3
                                0145  2133
                     AA000000   0145  2134 7$:      .LONG   ^XAA000000
                     55000000   0149  2135          .LONG   ^X55000000
                     33000000   014D  2136          .LONG   ^X33000000
                     0F000000   0151  2137          .LONG   ^X0F000000
                     00000000   0155  2138          .LONG   ^X00000000
                     00000000   0159  2139          .LONG   ^X00000000
                                015D  2140
                                015D  2141 BEGIN_CPU_DATA
                                0002  2142
                                0002  2143 DCS_DATA
                                0001  2144
U 00, 7700,C800,0364,0300,0470,1801 0001  2145 ;% 00   NOP                                         ; GET CPU UP TO SPEED
U 01, 5700,C860,0364,0300,0470,1802 000C  2149 ;% 01   M[TEMP0]_RDM                                ; MTEMP0 = TEST PATTERN
U 02, 6700,4800,4F47,0300,0470,1803 0017  2153 ;% 02   MSRC/TEMP0,ROT/CLR1BM,ALPCTL/WX_S.Q_0,RDM_WB    ; WHREG = SBUS DATA
U 03, 7300,C800,0364,0300,0470,1804 0022  2157 ;% 03   STOP.CLOCK                                   ; HALT
                                002D  2161
                                002D  2162 RTEMP_DATA
                                0001  2163
```

```
                    00000000  0001  2164          .LONG   0       ; INITIAL VALUE FOR MTEMP0
                              0005  2165
                              0005  2166 END_CPU_DATA
                              015D  2167
                              015D  2168 END_TEST_DATA
                              015D
                              015D        ;-------------------------------------------------------------------------
                              015D  2169
                              015D  2170 ENDTEST
```

```
                        002F        .SBTTL  "TEST   047     Rotator Test 21 Zero Extended Literal Test
                        002F  2172 ;*******************************************************************************
                        002F  2173 ;++
                        002F  2174 ;
                        002F  2175 ; FUNCTIONAL DESCRIPTION
                        002F  2176 ;
                        002F  2177 ;       This test examines the rotator functions that extract and zero extend
                        002F  2178 ;       the nine bit literal field.
                        002F  2179 ;
                        002F  2180 ; TEST ALGORITHM
                        002F  2181 ;
                        002F  2182 ;       This test is divided into two subtests. Subtest 1 examines the ZLITPL
                        002F  2183 ;       micro-order while Subtest 2 examines the ZLIT0 micro-order. The test
                        002F  2184 ;       algorithm is the same for both subtests.
                        002F  2185 ;
                        002F  2186 ;       1. Setup a loop of six iterations.
                        002F  2187 ;       2. Generate a stuck at zero or one test pattern.
                        002F  2188 ;       3. Load the literal field of the micro-word for the current subtest
                        002F  2189 ;          with the lower nine bits of the test pattern.
                        002F  2190 ;       4. Load the modified test micro-word into DCS.
                        002F  2191 ;       5. Execute the following micro-sequence:
                        002F  2192 ;                1. Zero the PLATCH.
                        002F  2193 ;                2. Execute the extract and read result into the RDM's WHREG.
                        002F  2194 ;       6. Check the result and signal an error if wrong.
                        002F  2195 ;       7. Loop back until done.
                        002F  2196 ;
                        002F  2197 ; TEST DATA
                        002F  2198 ;
                        002F  2199 ;       The following table gives the expected data for each subtest.
                        002F  2200 ;
                        002F  2201 ;             Test Pattern          Expected SBUS Data
                        002F  2202 ;
                        002F  2203 ;               AAAAAAAA                 000000AA
                        002F  2204 ;               55555555                 00000155
                        002F  2205 ;               33333333                 00000133
                        002F  2206 ;               0F0F0F0F                 0000010F
                        002F  2207 ;               00FF00FF                 000000FF
                        002F  2208 ;               0000FFFF                 000001FF
                        002F  2209 ;
                        002F  2210 ; LOGICAL DESCRIPTION
                        002F  2211 ;
                        002F  2212 ;       The logic involved is internal to the SRM gate array. SRM in turn
                        002F  2213 ;       gets some of its control signals from the SRK gate array.
                        002F  2214 ;
                        002F  2215 ; ASSUMPTIONS
                        002F  2216 ;
                        002F  2217 ;       This test requires an operational MBUS, SBUS, WBUS, and the ALPCTL
                        002F  2218 ;       special function WMUX <- BMUX.
                        002F  2219 ;
                        002F  2220 ; ERROR DESCRIPTION
                        002F  2221 ;
                        002F  2222 ;       On error this test will call out the SRM and SRK gate arrays.
                        002F  2223 ;       The following data will also be typed out:
                        002F  2224 ;
                        002F  2225 ;       EXPECTED SBUS DATA
```

```
                        002F  2226 ;       RECEIVED SBUS DATA
                        002F  2227 ;       LOOP COUNT
                        002F  2228 ;
                        002F  2229 ;--
                        002F  2230 ;*******************************************************************************
                        002F  2231
                        002F  2232         SUBTEST         1
                        0032
                        0032       ;////////////////////////////////////////////////////////////////////////////
                        0032  2233 ;+
                        0032  2234 ; Test the ROT/ZLITPL micro-order.
                        0032  2235 ;-
                        0032  2236         INITIALIZE                              ; INTIALIZE THE CPU
                        0034  2237         LOOP            I,1,6                   ; LOOP FOR 6 SA01 PATTERNS
                        003B  2238         SA01PAT         I,1$                    ; GENERATE A SA01 PATTERN
                        0043  2239         LDFIELD         1$,,W,2$,31,39          ; LOAD LITRL FIELD
                        0054  2240         LOADDCS         2$,,2,1                 ; LOAD MICRO-WORD INTO DCS
                        005B  2241         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        005D  2242         FETCH           0                       ; EXECUTE THE ZLITPL FUNCTION
                        0062  2243         BRSTCLK         3                       ; AND READ RESULT INTO WHREG
                        0066  2244         CMPREG          WHREG,4$,I,L            ; IS THE RESULT CORRECT?
                        006F  2245         IFERROR         ,<<SRM><SRK>>           ; IF NOT, SIGNAL AN ERROR
                        0076  2246         ENDLOOP         I                       ; CONTINUE LOOP
                        0079  2247
                        0079  2248         SUBTEST         2
                        007C
                        007C       ;////////////////////////////////////////////////////////////////////////////
                        007C  2249 ;+
                        007C  2250 ; Test the ROT/ZLIT0 micro-order.
                        007C  2251 ;-
                        007C  2252         LOOP            I,1,6                   ; LOOP FOR 6 SA01 PATTERNS
                        0083  2253         SA01PAT         I,1$                    ; GENERATE A SA01 PATTERN
                        008B  2254         LDFIELD         1$,,W,3$,31,39          ; LOAD LITRL FIELD
                        009C  2255         LOADDCS         3$,,2,1                 ; LOAD TEST MICRO-WORD INTO DCS
                        00A3  2256         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        00A5  2257         FETCH           0                       ; EXECUTE THE ZLIT0 FUNCTION
                        00AA  2258         BRSTCLK         3                       ; AND READ RESULT INTO WHREG
                        00AE  2259         CMPREG          WHREG,4$,I,L            ; IS THE RESULT CORRECT?
                        00B7  2260         IFERROR         ,<<SRM><SRK>>           ; IF NOT, SIGNAL AN ERROR
                        00BE  2261         ENDLOOP         I                       ; CONTINUE LOOP
                        00C1  2262         SKIP                                    ; END OF TEST
                        00C8  2263
                        00C8  2264 BEGIN_TEST_DATA
                        00C8
                        00C8       ;-------------------------------------------------------------------------
                        00C8  2265
                00000000  00C8  2266 1$:     .LONG   0       ; LOCATION TO HOLD SA01 PATTERN
                        00CC  2267
                        00CC  2268 2$:
U 02, 6700,9800,AF47,0355,0470,1803  00CC  2269 ;% 02   LIT/LITRL,LITRL/0AA,ROT/ZLITPL,ALPCTL/WX_S.Q_0,RDM_WB    ; SUBTEST 1
                        00D7  2273 3$:
U 02, 6700,9800,C347,0355,0470,1803  00D7  2274 ;% 02   LIT/LITRL,LITRL/0AA,ROT/ZLIT0,ALPCTL/WX_S.Q_0,RDM_WB     ; SUBTEST 2
                        00E2  2278
                        00E2  2279 ; Expected SBUS data
                        00E2  2280
```

```
                    000000AA  00E2  2281 4$:     .LONG   ^X000000AA
                    00000155  00E6  2282         .LONG   ^X00000155
                    00000133  00EA  2283         .LONG   ^X00000133
                    0000010F  00EE  2284         .LONG   ^X0000010F
                    000000FF  00F2  2285         .LONG   ^X000000FF
                    000001FF  00F6  2286         .LONG   ^X000001FF
                              00FA  2287
                              00FA  2288 BEGIN_CPU_DATA
                              0002  2289
                              0002  2290 DCS_DATA
                              0001  2291
U 00, 7700,C800,0364,0300,0470,1801   0001  2292 ;% 00   NOP                                                    ; CPU TO SPEED
U 01, 7700,1800,EF64,0300,0470,1802   000C  2296 ;% 01   PL_[0]                                                 ; ZERO PLATCH
U 02, 6700,9800,AF47,0355,0470,1803   0017  2300 ;% 02   LIT/LITRL,LITRL/0AA,ROT/ZLITPL,ALPCTL/WX_S.Q_0,RDM_WB  ; WHREG = SBUS
U 03, 7300,C800,0364,0300,0470,1804   0022  2304 ;% 03   STOP.CLOCK                                            ; HALT
                              002D  2308
                              002D  2309 END_CPU_DATA
                              00FA  2310
                              00FA  2311 END_TEST_DATA
                              00FA
                              00FA       ;-------------------------------------------------------------------------
                              00FA  2312
                              00FA  2313 ENDTEST
```

```
                002E        .SBTTL  "TEST   048     Rotator Test 22 One Extended Literal Test
                002E  2315 ;*******************************************************************************
                002E  2316 ;++
                002E  2317 ;
                002E  2318 ; FUNCTIONAL DESCRIPTION
                002E  2319 ;
                002E  2320 ;       This test examines the rotator functions that extract and one extend
                002E  2321 ;       the nine bit literal field.
                002E  2322 ;
                002E  2323 ; TEST ALGORITHM
                002E  2324 ;
                002E  2325 ;       This test is divided into four subtests. The first subtest examines
                002E  2326 ;       the OLIT0 micro-order, the second subtest examines the OLIT0.PL micro-
                002E  2327 ;       order, the third subtest examines the OLIT0.SL micro-order and the
                002E  2328 ;       fourth subtest examines the OLIT0.PL43 micro-order.
                002E  2329 ;
                002E  2330 ;       The following algorithm is used for each subtest:
                002E  2331 ;
                002E  2332 ;       1. Set up a loop of six iterations.
                002E  2333 ;       2. Generate a stuck at zero or one test pattern.
                002E  2334 ;       3. Load the literal field of the test micro-word for the current
                002E  2335 ;          subtest with the lower nine bits of the test pattern.
                002E  2336 ;       4. Load the modified micro-word into DCS.
                002E  2337 ;       5. Execute the following micro-sequence:
                002E  2338 ;                1. Perform the extract and read result into the RDM's WHREG.
                002E  2339 ;       6. Check the result and signal an error if wrong.
                002E  2340 ;       7. Loop back until done.
                002E  2341 ;
                002E  2342 ; TEST DATA
                002E  2343 ;
                002E  2344 ;       The following table gives the expected data for each subtest.
                002E  2345 ;
                002E  2346 ;             Test Pattern          Expected SBUS Data
                002E  2347 ;
                002E  2348 ;               AAAAAAAA                FFFFFEAA
                002E  2349 ;               55555555                FFFFFF55
                002E  2350 ;               33333333                FFFFFF33
                002E  2351 ;               0F0F0F0F                FFFFFF0F
                002E  2352 ;               00FF00FF                FFFFFEFF
                002E  2353 ;               0000FFFF                FFFFFFFF
                002E  2354 ;
                002E  2355 ; LOGICAL DESCRIPTION
                002E  2356 ;
                002E  2357 ;       The logic involved is internal to the SRM gate array. SRM in turn
                002E  2358 ;       gets some of its control signals from the SRK gate array.
                002E  2359 ;
                002E  2360 ; ASSUMPTIONS
                002E  2361 ;
                002E  2362 ;       This test requires an operational MBUS, SBUS, WBUS, and ALPCTL special
                002E  2363 ;       function WMUX <- BMUX.
                002E  2364 ;
                002E  2365 ; ERROR DESCRIPTION
                002E  2366 ;
                002E  2367 ;       On error this test will call out the SRM and SRK gate arrays.
                002E  2368 ;       The following data will also be typed out:
```

```
                        002E  2369 ;
                        002E  2370 ;       EXPECTED SBUS DATA
                        002E  2371 ;       RECEIVED SBUS DATA
                        002E  2372 ;       LOOP COUNT
                        002E  2373 ;
                        002E  2374 ;--
                        002E  2375 ;*****************************************************************************
                        002E  2376
                        002E  2377         SUBTEST         1
                        0031
                        0031       ;////////////////////////////////////////////////////////////////////////////
                        0031  2378 ;+
                        0031  2379 ; Check the ROT/OLIT0 micro-order.
                        0031  2380 ;-
                        0031  2381         INITIALIZE                              ; INITIALIZE THE CPU
                        0033  2382         LOOP            I,1,6                   ; LOOP FOR 6 SA01 PATTERNS
                        003A  2383         SA01PAT         I,1$                    ; GENERATE A SA01 PATTERN
                        0042  2384         LDFIELD         1$,,W,2$,31,39          ; LOAD THE LITRL FIELD
                        0053  2385         LOADDCS         2$,,1,1                 ; LOAD TEST MICRO-WORD INTO DCS
                        005A  2386         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        005C  2387         FETCH           0                       ; EXECUTE THE OLIT0 FUNCTION
                        0061  2388         BRSTCLK         2                       ; AND READ RESULT INTO WHREG
                        0065  2389         CMPREG          WHREG,6$,I,L            ; IS THE RESULT CORRECT?
                        006E  2390         IFERROR         ,<<SRM><SRK>>           ; IF NOT, SIGNAL AN ERROR
                        0075  2391         ENDLOOP         I                       ; CONTINUE LOOP
                        0078  2392
                        0078  2393         SUBTEST         2
                        007B
                        007B       ;////////////////////////////////////////////////////////////////////////////
                        007B  2394 ;+
                        007B  2395 ; Check the ROT/OLIT0.PL micro-order.
                        007B  2396 ;-
                        007B  2397         LOOP            I,1,6                   ; LOOP FOR 6 SA01 PATTERNS
                        0082  2398         SA01PAT         I,1$                    ; GENERATE A SA01 PATTERN
                        008A  2399         LDFIELD         1$,,W,3$,31,39          ; LOAD THE LITRL FIELD
                        009B  2400         LOADDCS         3$,,1,1                 ; LOAD TEST MICRO-WORD INTO DCS
                        00A2  2401         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        00A4  2402         FETCH           0                       ; EXECUTE THE OLIT0.PL FUNCTION
                        00A9  2403         BRSTCLK         2                       ; AND READ RESULT INTO WHREG
                        00AD  2404         CMPREG          WHREG,6$,I,L            ; IS THE RESULT CORRECT?
                        00B6  2405         IFERROR         ,<<SRM><SRK>>           ; IF NOT, SIGNAL AN ERROR
                        00BD  2406         ENDLOOP         I                       ; CONTINUE LOOP
                        00C0  2407
                        00C0  2408         SUBTEST         3
                        00C3
                        00C3       ;////////////////////////////////////////////////////////////////////////////
                        00C3  2409 ;+
                        00C3  2410 ; Check ROT/OLIT0.SL micro-order.
                        00C3  2411 ;-
                        00C3  2412         LOOP            I,1,6                   ; LOOP FOR 6 SA01 PATTERNS
                        00CA  2413         SA01PAT         I,1$                    ; GENERATE A SA01 PATTERN
                        00D2  2414         LDFIELD         1$,,W,4$,31,39          ; LOAD THE LITRL FIELD
                        00E3  2415         LOADDCS         4$,,1,1                 ; LOAD TEST MICRO-WORD INTO DCS
                        00EA  2416         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        00EC  2417         FETCH           0                       ; EXECUTE THE OLIT0.SL FUNCTION
```

```
                        00F1  2418          BRSTCLK         2                               ; AND READ RESULT INTO WHREG
                        00F5  2419          CMPREG          WHREG,6$,I,L                    ; IS THE RESULT CORRECT?
                        00FE  2420          IFERROR         ,<<SRM><SRK>>                   ; IF NOT, SIGNAL AN ERROR
                        0105  2421          ENDLOOP         I                               ; CONTINUE LOOP
                        0108  2422
                        0108  2423          SUBTEST         4
                        010B
                        010B          ;//////////////////////////////////////////////////////////////////////////////
                        010B  2424 ;+
                        010B  2425 ; Check ROT/OLIT0.PL43 micro-order.
                        010B  2426 ;-
                        010B  2427          LOOP            I,1,6                           ; LOOP FOR 6 SA01 PATTERNS
                        0112  2428          SA01PAT         I,1$                            ; GENERATE A SA01 PATTERN
                        011A  2429          LDFIELD         1$,,W,5$,31,39                  ; LOAD THE LITRL FIELD
                        012B  2430          LOADDCS         5$,,1,1                         ; LOAD TEST MICRO-WORD INTO DCS
                        0132  2431          ERRLOOP                                         ; LOOP ON ERROR FROM HERE
                        0134  2432          FETCH           0                               ; EXECUTE OLIT0.PL43 FUNCTION
                        0139  2433          BRSTCLK         2                               ; AND READ RESULT INTO WHREG
                        013D  2434          CMPREG          WHREG,6$,I,L                    ; IS THE RESULT CORRECT?
                        0146  2435          IFERROR         ,<<SRM><SRK>>                   ; IF NOT, SIGNAL AN ERROR
                        014D  2436          ENDLOOP         I                               ; CONTINUE LOOP
                        0150  2437          SKIP                                            ; END OF TEST
                        0157  2438
                        0157  2439 BEGIN_TEST_DATA
                        0157
                        0157          ;-------------------------------------------------------------------------
                        0157  2440
                00000000  0157  2441 1$:      .LONG   0       ; LOCATION TO HOLD SA01 PATTERN
                        015B  2442
                        015B  2443 2$:
U 01, 6700,5800,E347,0355,0470,1802     015B  2444 ;% 01   LIT/LITRL,LITRL/0AA,ROT/OLIT0,ALPCTL/WX_S.Q_0,RDM_WB          ;SUBTEST 1
                        0166  2448 3$:
U 01, 6700,5800,EF47,0355,0470,1802     0166  2449 ;% 01   LIT/LITRL,LITRL/0AA,ROT/OLIT0.PL_LIT,ALPCTL/WX_S.Q_0,RDM_WB   ;SUBTEST 2
                        0171  2453 4$:
U 01, 6700,5800,F747,0355,0470,1802     0171  2454 ;% 01   LIT/LITRL,LITRL/0AA,ROT/OLIT0.SL_LIT,ALPCTL/WX_S.Q_0,RDM_WB   ;SUBTEST 3
                        017C  2458 5$:
U 01, 6700,1800,FF47,0355,0470,1802     017C  2459 ;% 01   LIT/LITRL,LITRL/0AA,ROT/OLIT0.PL43_WB,ALPCTL/WX_S.Q_0,RDM_WB ;SUBTEST 4
                        0187  2463
                        0187  2464 ; Expected SBUS Data
                        0187  2465
                FFFFFEAA  0187  2466 6$:      .LONG   ^XFFFFFEAA
                FFFFFF55  018B  2467          .LONG   ^XFFFFFF55
                FFFFFF33  018F  2468          .LONG   ^XFFFFFF33
                FFFFFF0F  0193  2469          .LONG   ^XFFFFFF0F
                FFFFFEFF  0197  2470          .LONG   ^XFFFFFEFF
                FFFFFFFF  019B  2471          .LONG   ^XFFFFFFFF
                        019F  2472
                        019F  2473 BEGIN_CPU_DATA
                        0002  2474
                        0002  2475 DCS_DATA
                        0001  2476
U 00, 7700,C800,0364,0300,0470,1801     0001  2477 ;% 00   NOP                                                   ; CPU TO SPEED
U 01, 6700,5800,E347,0355,0470,1802     000C  2481 ;% 01   LIT/LITRL,LITRL/0AA,ROT/OLIT0,ALPCTL/WX_S.Q_0,RDM_WB     ; WHREG = SBUS
U 02, 7300,4800,0364,0300,0470,1803     0017  2485 ;% 02   STOP.CLOCK                                             ; HALT
                        0022  2489
```

```
                        0022  2490 END_CPU_DATA
                        019F  2491
                        019F  2492 END_TEST_DATA
                        019F
                        019F      ;-----------------------------------------------------------------------------
                        019F  2493
                        019F  2494 ENDTEST
```

```
0032        .SBTTL  "TEST   049     Rotator Test 23 Right Rotation of RBUS Test 1
0032  2496 ;*******************************************************************************
0032  2497 ;++
0032  2498 ;
0032  2499 ; FUNCTIONAL DESCRIPTION
0032  2500 ;
0032  2501 ;       The test examines the rotator's ability to do right circular rotation
0032  2502 ;       of the data on the RBUS. It is implemented in two subtests. The first
0032  2503 ;       subtest examines the second level shifter inside the ALP gate array
0032  2504 ;       while the second subtest examines the byte rotator inside the SRM chip.
0032  2505 ;
0032  2506 ; TEST ALGORITHM
0032  2507 ;
0032  2508 ;       Subtest 1
0032  2509 ;
0032  2510 ;       This subtest examines the second level shifter inside the ALP chip.
0032  2511 ;
0032  2512 ;        1. Load the K index with 1.
0032  2513 ;        2. Start signature analysis.
0032  2514 ;        3. Set up a loop of 4 iterations on I.
0032  2515 ;        4. Prepare the micro-instruction at 10$ to load the PLATCH with a test
0032  2516 ;           value from the table at 3$ indexed by I.
0032  2517 ;        5. Load the test micro-instruction into DCS.
0032  2518 ;        6. Set up a loop of 6 iterations on J.
0032  2519 ;        7. Based on the value of J, generate a standard stuck at zero or one
0032  2520 ;           test pattern and place it into 1$.
0032  2521 ;        8. Load the test pattern from 1$ into WDREG.
0032  2522 ;        9. Loop on error from here.
0032  2523 ;       10. Execute the micro-code to:
0032  2524 ;               1. Load the PLATCH with the value given in step 4.
0032  2525 ;               2. Load RTEMP0 with the data in the WDREG.
0032  2526 ;               3. Execute a ROT/RR.RR.P micro-order and read the result back
0032  2527 ;                  to the WHREG.
0032  2528 ;       11. Compare the data in the WHREG with the expected data from the table
0032  2529 ;           at 2$ indexed by K.
0032  2530 ;       12. If the expected and received data is not the same, issue an error.
0032  2531 ;       13. Increment the index K.
0032  2532 ;       14. Loop back to step 6 until all six patterns have been tried.
0032  2533 ;       15. Loop back to step 3 until all four PLATCH values have been tried.
0032  2534 ;       16. Stop signature analysis.
0032  2535 ;
0032  2536 ;       Subtest 2
0032  2537 ;
0032  2538 ;       This subtest examines the first level shifter inside the SRM chip.
0032  2539 ;
0032  2540 ;        1. Load the K index with 1.
0032  2541 ;        2. Start signature analysis.
0032  2542 ;        3. Set up a loop of 7 iterations on I.
0032  2543 ;        4. Prepare the micro-instruction at 10$ to load the PLATCH with a test
0032  2544 ;           value from the table at 5$ indexed by I.
0032  2545 ;        5. Load the test micro-instruction into DCS.
0032  2546 ;        6. Set up a loop of 6 iterations on J.
0032  2547 ;        7. Based on the value of J, generate a standard stuck at zero or one
0032  2548 ;           test pattern and place it into 1$.
0032  2549 ;        8. Load the test pattern from 1$ into WDREG.
```

```
0032  2550 ;       9. Loop on error from here.
0032  2551 ;      10. Execute the micro-code to:
0032  2552 ;              1. Load the PLATCH with the value given in step 4.
0032  2553 ;              2. Load RTEMP0 with the data in the WDREG.
0032  2554 ;              3. Execute a ROT/RR.RR.P micro-order and read the result back
0032  2555 ;                 to the WHREG.
0032  2556 ;      11. Compare the data in the WHREG with the expected data from the table
0032  2557 ;          at 4$ indexed by K.
0032  2558 ;      12. If the expected and received data is not the same, issue an error.
0032  2559 ;      13. Increment the index K.
0032  2560 ;      14. Loop back to step 6 until all 6 patterns have been tried.
0032  2561 ;      15. Loop back to step 3 until all 7 PLATCH values have been tried.
0032  2562 ;      16. Stop signature analysis.
0032  2563 ;
0032  2564 ; TEST DATA
0032  2565 ;
0032  2566 ;       The data patterns input to the rotator over the RBUS are the standard
0032  2567 ;       stuck at zero or one test pattterns used in most register tests.
0032  2568 ;       The expected data consists of the same data rotated right the proper
0032  2569 ;       number of bits. By using these data patterns it is possible to find any
0032  2570 ;       stuck bit in the rotator data path as well as any bit tied to any other
0032  2571 ;       bit, even after it has undergone a rotation.
0032  2572 ;
0032  2573 ;       Subtest 1
0032  2574 ;
0032  2575 ;       Index I = PLATCH test data being used.
0032  2576 ;       Index J = Stuck at zero or one test pattern being used.
0032  2577 ;       Index K = Index to expected data table.
0032  2578 ;
0032  2579 ;       I  J  K          PLATCH          RBUS          WBUS
0032  2580 ;
0032  2581 ;       01 01 01         00000000        AAAAAAAA      AAAAAAAA
0032  2582 ;       01 02 02         00000000        55555555      55555555
0032  2583 ;       01 03 03         00000000        33333333      33333333
0032  2584 ;       01 04 04         00000000        0F0F0F0F      0F0F0F0F
0032  2585 ;       01 05 05         00000000        00FF00FF      00FF00FF
0032  2586 ;       01 06 06         00000000        0000FFFF      0000FFFF
0032  2587 ;       02 01 07         00000001        AAAAAAAA      55555555
0032  2588 ;       02 02 08         00000001        55555555      AAAAAAAA
0032  2589 ;       02 03 09         00000001        33333333      99999999
0032  2590 ;       02 04 0A         00000001        0F0F0F0F      87878787
0032  2591 ;       02 05 0B         00000001        00FF00FF      807F807F
0032  2592 ;       02 06 0C         00000001        0000FFFF      80007FFF
0032  2593 ;       03 01 0D         00000002        AAAAAAAA      AAAAAAAA
0032  2594 ;       03 02 0E         00000002        55555555      55555555
0032  2595 ;       03 03 0F         00000002        33333333      CCCCCCCC
0032  2596 ;       03 04 10         00000002        0F0F0F0F      C3C3C3C3
0032  2597 ;       03 05 11         00000002        00FF00FF      C03FC03F
0032  2598 ;       03 06 12         00000002        0000FFFF      C0003FFF
0032  2599 ;       04 01 13         00000003        AAAAAAAA      55555555
0032  2600 ;       04 02 14         00000003        55555555      AAAAAAAA
0032  2601 ;       04 03 15         00000003        33333333      66666666
0032  2602 ;       04 04 16         00000003        0F0F0F0F      E1E1E1E1
0032  2603 ;       04 05 16         00000003        00FF00FF      E01FE01F
0032  2604 ;       04 06 17         00000003        0000FFFF      E0001FFF
```

```
0032  2605 ;
0032  2606 ;       Subtest 2
0032  2607 ;
0032  2608 ;       Index I = PLATCH test data being used.
0032  2609 ;       Index J = Stuck at zero or one test pattern being used.
0032  2610 ;       Index K = Index to expected data table.
0032  2611 ;
0032  2612 ;       I  J  K          PLATCH          RBUS          WBUS
0032  2613 ;
0032  2614 ;       01 01 01        00000004        AAAAAAAA      AAAAAAAA
0032  2615 ;       01 02 02        00000004        55555555      55555555
0032  2616 ;       01 03 03        00000004        33333333      33333333
0032  2617 ;       01 04 04        00000004        0F0F0F0F      F0F0F0F0
0032  2618 ;       01 05 05        00000004        00FF00FF      F00FF00F
0032  2619 ;       01 06 06        00000004        0000FFFF      F0000FFF
0032  2620 ;       02 01 07        00000008        AAAAAAAA      AAAAAAAA
0032  2621 ;       02 02 08        00000008        55555555      55555555
0032  2622 ;       02 03 09        00000008        33333333      33333333
0032  2623 ;       02 04 0A        00000008        0F0F0F0F      0F0F0F0F
0032  2624 ;       02 05 0B        00000008        00FF00FF      FF00FF00
0032  2625 ;       02 06 0C        00000008        0000FFFF      FF0000FF
0032  2626 ;       03 01 0D        0000000C        AAAAAAAA      AAAAAAAA
0032  2627 ;       03 02 0E        0000000C        55555555      55555555
0032  2628 ;       03 03 0F        0000000C        33333333      33333333
0032  2629 ;       03 04 10        0000000C        0F0F0F0F      F0F0F0F0
0032  2630 ;       03 05 11        0000000C        00FF00FF      0FF00FF0
0032  2631 ;       03 06 12        0000000C        0000FFFF      FFF0000F
0032  2632 ;       04 01 13        00000010        AAAAAAAA      AAAAAAAA
0032  2633 ;       04 02 14        00000010        55555555      55555555
0032  2634 ;       04 03 15        00000010        33333333      33333333
0032  2635 ;       04 04 16        00000010        0F0F0F0F      0F0F0F0F
0032  2636 ;       04 05 17        00000010        00FF00FF      00FF00FF
0032  2637 ;       04 06 18        00000010        0000FFFF      FFFF0000
0032  2638 ;       05 01 19        00000014        AAAAAAAA      AAAAAAAA
0032  2639 ;       05 02 1A        00000014        55555555      55555555
0032  2640 ;       05 03 1B        00000014        33333333      33333333
0032  2641 ;       05 04 1C        00000014        0F0F0F0F      F0F0F0F0
0032  2642 ;       05 05 1D        00000014        00FF00FF      F00FF00F
0032  2643 ;       05 06 1E        00000014        0000FFFF      0FFFF000
0032  2644 ;       06 01 1F        00000018        AAAAAAAA      AAAAAAAA
0032  2645 ;       06 02 20        00000018        55555555      55555555
0032  2646 ;       06 03 21        00000018        33333333      33333333
0032  2647 ;       06 04 22        00000018        0F0F0F0F      0F0F0F0F
0032  2648 ;       06 05 23        00000018        00FF00FF      FF00FF00
0032  2649 ;       06 06 24        00000018        0000FFFF      00FFFF00
0032  2650 ;       07 01 25        0000001C        AAAAAAAA      AAAAAAAA
0032  2651 ;       07 02 26        0000001C        55555555      55555555
0032  2652 ;       07 03 27        0000001C        33333333      33333333
0032  2653 ;       07 04 28        0000001C        0F0F0F0F      F0F0F0F0
0032  2654 ;       07 05 29        0000001C        00FF00FF      0FF00FF0
0032  2655 ;       07 06 2A        0000001C        0000FFFF      000FFFF0
0032  2656 ;
0032  2657 ; LOGIC DESCRIPTION
0032  2658 ;
0032  2659 ;       The right rotation data path is in the SRM and ALP gate arrays on the
```

```
                        0032  2660 ;       DPM module. The right rotation function is decoded from the ROT field
                        0032  2661 ;       of the micro-word by the SRK and ALK gate arrays also on DPM.
                        0032  2662 ;       The decoded ROT field generates the control signals SHF1L, SHF0L, PRI1
                        0032  2663 ;       and PRI0 that control the SRM and ALP.
                        0032  2664 ;
                        0032  2665 ; ASSUMPTIONS
                        0032  2666 ;
                        0032  2667 ;       This test assumes the scratchpad RTEMP0 is working as well as the RBUS
                        0032  2668 ;       SBUS, WBUS and the rotator's PLATCH.
                        0032  2669 ;
                        0032  2670 ; ERROR DESCRIPTION
                        0032  2671 ;
                        0032  2672 ;       On error Subtest 1 will call out the ALP,SRM,SRK and ALK gate arrays.
                        0032  2673 ;       On error Subtest 2 will call out the SRM and SRK gate arrays.
                        0032  2674 ;
                        0032  2675 ;       On error both subtests will type out the following information:
                        0032  2676 ;
                        0032  2677 ;       EXPECTED DATA
                        0032  2678 ;       RECEIVED DATA
                        0032  2679 ;       LOOP COUNT I
                        0032  2680 ;       LOOP COUNT J
                        0032  2681 ;       INDEX VALUE K
                        0032  2682 ;
                        0032  2683 ;--
                        0032  2684 ;*******************************************************************************
                        0032  2685
                        0032  2686         SUBTEST         1
                        0035
                        0035       ;////////////////////////////////////////////////////////////////////////////
                        0035  2687 ;*
                        0035  2688 ; Test the second level shifter.
                        0035  2689 ;-
                        0035  2690         INITIALIZE                                      ; INITIALIZE THE CPU
                        0037  2691         LDINDEX         K,1                             ; INITIALIZE THE INDEX K
                        003E  2692         LOOP            I,1,4                           ; LOOP ON I FOR 4 PLATCH VALUES
                        0045  2693         LDFIELD         3$,I,W,10$,31,39                ; LITRL FIELD GETS PLATCH VALUE
                        0056  2694         LOADDCS         10$,,1,1                        ; LOAD PL_[LITRL] INTO DCS
                        005D  2695         LOOP            J,1,6                           ; LOOP ON J FOR 6 TEST PATTERNS
                        0064  2696         SA01PAT         J,1$                            ; GENERATE A SA01 TEST PATTERN
                        006C  2697         LOADREG         WDREG,1$,,L                     ; LOAD TEST PATTERN INTO WDREG
                        0074  2698         ERRLOOP                                         ; LOOP ON ERROR FROM HERE
                        0076  2699         FETCH           0                               ; EXECUTE THE ROT/RR.RR.P ORDER
                        007B  2700         BRSTCLK         4                               ; AND WRITE RESULT TO WHREG
                        007F  2701         CMPREG          WHREG,2$,K,L                    ; IS WHREG CORRECT?
                        0088  2702         IFERROR         ,<<ALP><SRM><SRK><ALK>>         ; IF NOT, SIGNAL AN ERROR
                        0091  2703         INCINDEX        K                               ; INCREMENT INDEX K
                        0094  2704         ENDLOOP         J                               ; CONTINUE SA01 PATTERNS
                        0097  2705         ENDLOOP         I                               ; CONTINUE PLATCH VALUES
                        009A  2706
                        009A  2707         SUBTEST         2
                        009D
                        009D       ;////////////////////////////////////////////////////////////////////////////
                        009D  2708 ;*
                        009D  2709 ; Test the rotation logic inside and across the SRM gate arrays.
                        009D  2710 ;-
```

```
                        009D  2711          LDINDEX         K,1                             ; INITIALIZE THE INDEX K
                        00A4  2712          LOOP            I,1,7                           ; LOOP ON I FOR 7 PLATCH VALUES
                        00AB  2713          LDFIELD         5$,I,W,10$,31,39                ; LOAD LITRL WITH PLATCH VALUE
                        00BC  2714          LOADDCS         10$,,1,1                        ; LOAD PL_[LITRL] INTO DCS
                        00C3  2715          LOOP            J,1,6                           ; LOOP ON J FOR 6 SA01 PATTERNS
                        00CA  2716          SA01PAT         J,1$                            ; GENERATE A SA01 TEST PATTERN
                        00D2  2717          LOADREG         WDREG,1$,,L                     ; LOAD SA01 PATTERN INTO WDREG
                        00DA  2718          ERRLOOP                                         ; LOOP ON ERROR FROM HERE
                        00DC  2719          FETCH           0                               ; EXECUTE THE ROT/RR.RR.P ORDER
                        00E1  2720          BRSTCLK         4                               ; AND WRITE RESULT INTO WHREG
                        00E5  2721          CMPREG          WHREG,4$,K,L                    ; IS WHREG CORRECT?
                        00EE  2722          IFERROR         ,<<SRM><SRK>>                   ; IF NOT, SIGNAL AN ERROR
                        00F5  2723          INCINDEX        K                               ; INCREMENT THE INDEX K
                        00F8  2724          ENDLOOP         J                               ; CONTINUE SA01 PATTERNS
                        00FB  2725          ENDLOOP         I                               ; CONTINUE PLATCH VALUES
                        00FE  2726          SKIP                                            ; SKIP TO NEXT TEST
                        0105  2727
                        0105  2728 BEGIN_TEST_DATA
                        0105
                        0105          ;-----------------------------------------------------------------------
                        0105  2729
                00000000  0105  2730 1$:      .LONG   0       ; LOCATION TO HOLD SA01PAT FOR WDREG
                        0109  2731
                        0109  2732 ; Expected data for Subtest 1
                        0109  2733 ; The numbers on the right indicate values for index K.
                        0109  2734
33333333 55555555 AAAAAAAA  0109  2735 2$:      .LONG   ^XAAAAAAAA,^X55555555,^X33333333        ; 01 - 03
0000FFFF 00FF00FF 0F0F0F0F  0115  2736          .LONG   ^X0F0F0F0F,^X00FF00FF,^X0000FFFF        ; 04 - 06
99999999 AAAAAAAA 55555555  0121  2737          .LONG   ^X55555555,^XAAAAAAAA,^X99999999        ; 07 - 09
80007FFF 807F807F 87878787  012D  2738          .LONG   ^X87878787,^X807F807F,^X80007FFF        ; 0A - 0C
CCCCCCCC 55555555 AAAAAAAA  0139  2739          .LONG   ^XAAAAAAAA,^X55555555,^XCCCCCCCC        ; 0D - 0F
C0003FFF C03FC03F C3C3C3C3  0145  2740          .LONG   ^XC3C3C3C3,^XC03FC03F,^XC0003FFF        ; 10 - 12
66666666 AAAAAAAA 55555555  0151  2741          .LONG   ^X55555555,^XAAAAAAAA,^X66666666        ; 13 - 15
E0001FFF E01FE01F E1E1E1E1  015D  2742          .LONG   ^XE1E1E1E1,^XE01FE01F,^XE0001FFF        ; 16 - 18
                        0169  2743
                        0169  2744 ; PLATCH values for Subtest 1
                        0169  2745
                    0000  0169  2746 3$:      .WORD   0       ; I = 1
                    0001  016B  2747          .WORD   1       ; I = 2
                    0002  016D  2748          .WORD   2       ; I = 3
                    0003  016F  2749          .WORD   3       ; I = 4
                        0171  2750
                        0171  2751 ; Expected Data for Subtest 2
                        0171  2752 ; The numbers on the right indicate values for index K.
                        0171  2753
33333333 55555555 AAAAAAAA  0171  2754 4$:      .LONG   ^XAAAAAAAA,^X55555555,^X33333333        ; 01 - 03
F0000FFF F00FF00F F0F0F0F0  017D  2755          .LONG   ^XF0F0F0F0,^XF00FF00F,^XF0000FFF        ; 04 - 06
33333333 55555555 AAAAAAAA  0189  2756          .LONG   ^XAAAAAAAA,^X55555555,^X33333333        ; 07 - 09
FF0000FF FF00FF00 0F0F0F0F  0195  2757          .LONG   ^X0F0F0F0F,^XFF00FF00,^XFF0000FF        ; 0A - 0C
33333333 55555555 AAAAAAAA  01A1  2758          .LONG   ^XAAAAAAAA,^X55555555,^X33333333        ; 0D - 0F
FFF0000F 0FF00FF0 F0F0F0F0  01AD  2759          .LONG   ^XF0F0F0F0,^X0FF00FF0,^XFFF0000F        ; 10 - 12
33333333 55555555 AAAAAAAA  01B9  2760          .LONG   ^XAAAAAAAA,^X55555555,^X33333333        ; 13 - 15
FFFF0000 00FF00FF 0F0F0F0F  01C5  2761          .LONG   ^X0F0F0F0F,^X00FF00FF,^XFFFF0000        ; 16 - 18
33333333 55555555 AAAAAAAA  01D1  2762          .LONG   ^XAAAAAAAA,^X55555555,^X33333333        ; 19 - 1B
0FFFF000 F00FF00F F0F0F0F0  01DD  2763          .LONG   ^XF0F0F0F0,^XF00FF00F,^X0FFFF000        ; 1C - 1E
```

```
         33333333 55555555 AAAAAAAA  01E9  2764          .LONG   ^XAAAAAAAA,^X55555555,^X33333333        ; 1F - 21
         00FFFF00 FF00FF00 0F0F0F0F  01F5  2765          .LONG   ^X0F0F0F0F,^XFF00FF00,^X00FFFF00        ; 22 - 24
         33333333 55555555 AAAAAAAA  0201  2766          .LONG   ^XAAAAAAAA,^X55555555,^X33333333        ; 25 - 27
         000FFFF0 0FF00FF0 F0F0F0F0  020D  2767          .LONG   ^XF0F0F0F0,^X0FF00FF0,^X000FFFF0        ; 28 - 2A
                                     0219  2768
                                     0219  2769 ; PLATCH values for Subtest 2
                                     0219  2770
                               0004  0219  2771 5$:      .WORD   4       ; I = 1
                               0008  021B  2772          .WORD   8       ; I = 2
                               000C  021D  2773          .WORD   12      ; I = 3
                               0010  021F  2774          .WORD   16      ; I = 4
                               0014  0221  2775          .WORD   20      ; I = 5
                               0018  0223  2776          .WORD   24      ; I = 6
                               001C  0225  2777          .WORD   28      ; I = 7
                                     0227  2778
                                     0227  2779 ; Microword to load the PLATCH
                                     0227  2780
                                     0227  2781 10$:
U 01, 7700,1800,EF64,0300,0470,1802  0227  2782 ;% 01   PL_[0]          ; PLATCH GETS THE CONTENTS OF THE LITRL FIELD
                                     0232  2786
                                     0232  2787 BEGIN_CPU_DATA
                                     0002  2788
                                     0002  2789 DCS_DATA
                                     0001  2790
U 00, 7700,C800,0364,0300,0470,1801  0001  2791 ;% 00   NOP
U 01, 7700,1800,EF64,0300,0470,1802  000C  2795 ;% 01   PL_[0]
U 02, 5700,0840,0364,0300,0470,1803  0017  2799 ;% 02   R[TEMP0]_RDM
U 03, 6700,8800,1B47,0300,0470,1804  0022  2803 ;% 03   RSRC/TEMP0,ROT/RR.RR.P,ALPCTL/WX_S.Q_0,RDM_WB
U 04, 7300,4800,0364,0300,0470,1805  002D  2807 ;% 04   STOP.CLOCK
                                     0038  2811
                                     0038  2812 RTEMP_DATA
                                     0001  2813
                           00000000  0001  2814          .LONG   0       ; INITIALIZES RTEMP0
                                     0005  2815
                                     0005  2816 END_CPU_DATA
                                     0232  2817
                                     0232  2818 END_TEST_DATA
                                     0232
                                     0232       ;-----------------------------------------------------------------------------
                                     0232  2819
                                     0232  2820 ENDTEST
```

```
0032        .SBTTL  "TEST   04A     Rotator Test 24 Right Rotation of RBUS Test 2
0032  2822 ;*******************************************************************************
0032  2823 ;++
0032  2824 ;
0032  2825 ; FUNCTIONAL DESCRIPTION
0032  2826 ;
0032  2827 ;       This test examines the rotator micro-orders RR.RR.PS and RR.RR.SIZ.
0032  2828 ;       The first micro-order rotates the data on the RBUS right by the number
0032  2829 ;       of bits specified from the sum of the contents of the PLATCH and the
0032  2830 ;       SLATCH. The second micro-order right rotates the data on the RBUS by
0032  2831 ;       the number of bits specified in the DTYPE field of the test micro-
0032  2832 ;       instruction.
0032  2833 ;
0032  2834 ; TEST ALGORITHM
0032  2835 ;
0032  2836 ;       This test is implemented in three subtests. The first subtest floats a
0032  2837 ;       zero through a field of ones in the PLATCH with a constant one loaded
0032  2838 ;       into the SLATCH. It then adds the two together and uses the sum to
0032  2839 ;       right rotate the data on the RBUS by that number of bits. The second
0032  2840 ;       subtest is identical to the first, except that the floating zero is
0032  2841 ;       run through the SLATCH with a constant one loaded into the PLATCH.
0032  2842 ;       The third subtest loads the DTYPE field of the test micro-instruction
0032  2843 ;       with the three data types BYTE, WORD and LONGWORD and then checks that
0032  2844 ;       the data on the RBUS is right rotated the correct number of bytes for
0032  2845 ;       each data type. The algorithms for the subtests are given below:
0032  2846 ;
0032  2847 ;       Subtests 1 and 2
0032  2848 ;
0032  2849 ;        1. Initialize the CPU. (Subtest 1 only)
0032  2850 ;        2. Load a constant 1 into SLATCH (Subtest 1) or PLATCH (Subtest 2).
0032  2851 ;        3. Load the micro-instruction to test the P + S latch rotation.
0032  2852 ;        4. Load the index K with 1.
0032  2853 ;        5. Start signature analysis.
0032  2854 ;        6. Prepare to loop on I for 5 latch patterns.
0032  2855 ;        7. Load the RSRC field of the test micro-instruction to source TEMP0.
0032  2856 ;        8. Prepare a loop on J for 6 data patterns.
0032  2857 ;        9. Generate a test pattern based on J.
0032  2858 ;       10. Load the test pattern into the WDREG.
0032  2859 ;       11. Loop on error from here.
0032  2860 ;       12. Execute the micro-code to:
0032  2861 ;               1. Load PLATCH with a pattern (Subtest 1) or 1 (Subtest 2).
0032  2862 ;               2. Load the SLATCH with 1 (Subtest 1) or a pattern (Subtest 2).
0032  2863 ;               3. Load RTEMP0 with the test data from WDREG.
0032  2864 ;               4. Execute the test instruction and read the result to WHREG.
0032  2865 ;       13. Compare the WHREG contents with the expected data from a table
0032  2866 ;           indexed by K.
0032  2867 ;       14. If WHREG does not equal the expected data, signal an error.
0032  2868 ;       15. Increment the index K.
0032  2869 ;       16. Loop back to step 8 for rest of the data patterns.
0032  2870 ;       17. Loop back to step 6 for rest of latch patterns.
0032  2871 ;       18. End signature analysis.
0032  2872 ;
0032  2873 ;       Subtest 3
0032  2874 ;
0032  2875 ;        1. Load the index K with 1.
```

```
0032  2876 ;        2. Start signature analysis.
0032  2877 ;        3. Prepare to loop on I for 3 data types.
0032  2878 ;        4. Load a data type (either BYTE, WORD or LONG) into the DTYPE field
0032  2879 ;           of the test micro-instruction from the table at 5$ indexed by I.
0032  2880 ;        5. Load the test micro-instruction for Subtest 3 into DCS.
0032  2881 ;        6. Prepare a loop on J for 6 data patterns.
0032  2882 ;        7. Generate a standard stuck at zero or one test pattern based on J.
0032  2883 ;        8. Load the test pattern into the WDREG.
0032  2884 ;        9. Loop on error from here.
0032  2885 ;       10. Execute the micro-code to:
0032  2886 ;               1. Load RTEMP0 with the test pattern from WDREG.
0032  2887 ;               2. Rotate RTEMP0 right by DTYPE and write the result to WHREG.
0032  2888 ;       11. Compare the data in WHREG with the expected data from the table at
0032  2889 ;           6$ indexed by K.
0032  2890 ;       12. If WHREG does not equal the expected data, signal an error.
0032  2891 ;       13. Increment the index K.
0032  2892 ;       14. Continue the loop on J for the remaining data patterns.
0032  2893 ;       15. Continue the loop on I for the remaining data types.
0032  2894 ;
0032  2895 ; TEST DATA
0032  2896 ;
0032  2897 ;       The test data for Subtests 1 and 2 consist of a floating 0 pattern
0032  2898 ;       through a field of ones. This gives a good test of the SRK ALU by
0032  2899 ;       causing carries to ripple across bits <0:4> of the SLATCH and PLATCH.
0032  2900 ;       Bit <5> of the SLATCH and PLATCH is used for sign only and does not
0032  2901 ;       affect the number of bits shifted.
0032  2902 ;
0032  2903 ;       The data for Subtest 3 uses the 3 data types that can be loaded into
0032  2904 ;       the DTYPE field of the test micro-instruction; BYTE, WORD and LONG.
0032  2905 ;
0032  2906 ; LOGIC DESCRIPTION
0032  2907 ;
0032  2908 ;       The logic used to execute the right rotation of data on the RBUS is
0032  2909 ;       internal to the SRM and ALP gate arrays on the DPM module. The ALP
0032  2910 ;       gate array contains the second level shifter which can rotate data on
0032  2911 ;       the RBUS by 0,1,2 or 3 bits. The SRM contains the first level shifter
0032  2912 ;       which can rotate data on the RBUS from 0 to 7 bytes at a time. The
0032  2913 ;       SRK and ALK gate arrays are responsible for controlling the actions of
0032  2914 ;       this two level rotator.
0032  2915 ;
0032  2916 ; ASSUMPTIONS
0032  2917 ;
0032  2918 ;       This test assumes that the scratchpads RTEMP0 and MTEMP1 are working as
0032  2919 ;       well as the RBUS, SBUS, WBUS and the rotator's PLATCH and SLATCH.
0032  2920 ;
0032  2921 ; ERROR DESCRIPTION
0032  2922 ;
0032  2923 ;       On error Subtests 1 and 2 call out the SRK,SRM,ALP and ALK gate arrays.
0032  2924 ;       On error Subtest 3 will call out the SRK and SRM gate arrays.
0032  2925 ;
0032  2926 ;       On error all subtests will also type out the following data:
0032  2927 ;
0032  2928 ;       EXPECTED DATA
0032  2929 ;       RECEIVED DATA
0032  2930 ;       LOOP COUNT I
```

```
                        0032  2931 ;       LOOP COUNT J
                        0032  2932 ;       INDEX VALUE K
                        0032  2933 ;
                        0032  2934 ;--
                        0032  2935 ;*******************************************************************************
                        0032  2936
                        0032  2937         SUBTEST         1
                        0035
                        0035       ;////////////////////////////////////////////////////////////////////////////
                        0035  2938 ;+
                        0035  2939 ; Tests PLATCH + SLATCH rotation with PLATCH = 30,29,27,23,15 and SLATCH = 1.
                        0035  2940 ;-
                        0035  2941         INITIALIZE                                  ; INITIALIZE THE CPU
                        0037  2942         LDFIELD         3$,,W,11$,31,39             ; PUT A 1 INTO THE LITRL FIELD
                        0048  2943         LOADDCS         11$,,2,1                    ; LOAD SL_[1] INTO DCS
                        004F  2944         LOADDCS         12$,,4,1                    ; LOAD ROT/RR.RR.PS TEST ORDER
                        0056  2945         LDINDEX         K,1                         ; INITIALIZE INDEX K
                        005D  2946         LOOP            I,1,5                       ; LOOP FOR 5 PLATCH PATTERNS
                        0064  2947         LDFIELD         2$,I,W,10$,31,39            ; LOAD PATTERN INTO LITRL FIELD
                        0075  2948         LOADDCS         10$,,1,1                    ; LOAD PL_[PATTERN] INTO DCS
                        007C  2949         LOOP            J,1,6                       ; LOOP FOR 6 SA01 DATA PATTERNS
                        0083  2950         SA01PAT         J,1$                        ; GENERATE A SA01 TEST PATTERN
                        008B  2951         LOADREG         WDREG,1$,,L                 ; LOAD PATTERN INTO WDREG
                        0093  2952         ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                        0095  2953         FETCH           0                           ; EXECUTE THE ROT/RR.RR.PS AND
                        009A  2954         BRSTCLK         5                           ; WRITE THE RESULT INTO WHREG
                        009E  2955         CMPREG          WHREG,4$,K,L                ; ROTATION EXECUTE CORRECTLY?
                        00A7  2956         IFERROR         ,<<SRK><SRM><ALP><ALK>>     ; IF NOT, SIGNAL AN ERROR
                        00B0  2957         INCINDEX        K                           ; INCREMENT INDEX K
                        00B3  2958         ENDLOOP         J                           ; CONTINUE SA01 PATTERNS
                        00B6  2959         ENDLOOP         I                           ; CONTINUE PLATCH PATTERNS
                        00B9  2960
                        00B9  2961         SUBTEST         2
                        00BC
                        00BC       ;////////////////////////////////////////////////////////////////////////////
                        00BC  2962 ;+
                        00BC  2963 ; Tests PLATCH + SLATCH rotation with SLATCH = 30,29,27,23,15 PLATCH = 1.
                        00BC  2964 ;-
                        00BC  2965         LDFIELD         3$,,W,10$,31,39             ; PUT A 1 INTO THE LITRL FIELD
                        00CD  2966         LOADDCS         10$,,1,1                    ; LOAD PL_[1] INTO DCS
                        00D4  2967         LDINDEX         K,1                         ; INITIALIZE THE INDEX K
                        00DB  2968         LOOP            I,1,5                       ; LOOP FOR 5 SLATCH PATTERNS
                        00E2  2969         LDFIELD         2$,I,W,11$,31,39            ; LOAD PATTERN INTO LITRL FIELD
                        00F3  2970         LOADDCS         11$,,2,1                    ; LOAD SL_[PATTERN] INTO DCS
                        00FA  2971         LOOP            J,1,6                       ; LOOP FOR 6 SA01 TEST PATTERNS
                        0101  2972         SA01PAT         J,1$                        ; GENERATE A SA01 TEST PATTERN
                        0109  2973         LOADREG         WDREG,1$,,L                 ; LOAD TEST PATTERN INTO WDREG
                        0111  2974         ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                        0113  2975         FETCH           0                           ; EXECUTE THE ROT/RR.RR.PS AND
                        0118  2976         BRSTCLK         5                           ; WRITE THE RESULT INTO WHREG
                        011C  2977         CMPREG          WHREG,4$,K,L                ; ROTATION EXECUTE CORRECTLY?
                        0125  2978         IFERROR         ,<<SRK><SRM><ALP><ALK>>     ; IF NOT, SIGNAL AN ERROR
                        012E  2979         INCINDEX        K                           ; INCREMENT THE INDEX K
                        0131  2980         ENDLOOP         J                           ; CONTINUE SA01 PATTERNS
                        0134  2981         ENDLOOP         I                           ; CONTINUE SLATCH PATTERNS
```

```
                        0137  2982
                        0137  2983          SUBTEST         3
                        013A
                        013A          ;////////////////////////////////////////////////////////////////////////////
                        013A  2984 ;+
                        013A  2985 ; Tests the ROT/RR.RR.SIZ micro-order for DTYPE values of BYTE, WORD and LONG.
                        013A  2986 ;-
                        013A  2987          LDINDEX         K,1                     ; INITIALIZE THE INDEX K
                        0141  2988          LOOP            I,1,3                   ; LOOP FOR 3 DTYPES
                        0148  2989          LDFIELD         5$,I,B,13$,40,41        ; LOAD THE DTYPE FIELD
                        0159  2990          LOADDCS         13$,,4,1                ; LOAD THE TEST INSTRUCTION
                        0160  2991          LOOP            J,1,6                   ; LOOP FOR 6 SA01 PATTERNS
                        0167  2992          SA01PAT         J,1$                    ; GENERATE A SA01 PATTERN
                        016F  2993          LOADREG         WDREG,1$,,L             ; LOAD THE PATTERN INTO WDREG
                        0177  2994          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        0179  2995          FETCH           0                       ; EXECUTE ROT/RR.RR.SIZ AND
                        017E  2996          BRSTCLK         5                       ; WRITE THE RESULT TO WHREG
                        0182  2997          CMPREG          WHREG,6$,K,L            ; ROTATION EXECUTE CORRECTLY?
                        018B  2998          IFERROR         ,<<SRK><SRM>>           ; IF NOT, SIGNAL AN ERROR
                        0192  2999          INCINDEX        K                       ; INCREMENT INDEX K
                        0195  3000          ENDLOOP         J                       ; CONTINUE SA01 PATTERNS
                        0198  3001          ENDLOOP         I                       ; CONTINUE DTYPES
                        019B  3002          SKIP                                    ; SKIP TO NEXT TEST
                        01A2  3003
                        01A2  3004 BEGIN_TEST_DATA
                        01A2
                        01A2          ;-----------------------------------------------------------------------
                        01A2  3005
                00000000  01A2  3006 1$:     .LONG   0       ; LOCATION TO HOLD SA01PAT FOR WDREG
                        01A6  3007
                        01A6  3008 ; Latch values for Subtests 1 and 2
                        01A6  3009
  000F 0017 001B 001D 001E  01A6  3010 2$:     .WORD   ^X1E,^X1D,^X1B,^X17,^X0F        ; FLOATING 0 PATTERN
                     0001  01B0  3011 3$:     .WORD   ^X01                            ; CONSTANT 1
                        01B2  3012
                        01B2  3013 ; Expected Data for Subtests 1 and 2
                        01B2  3014 ; The numbers to the right indicate LATCH and K index values.
                        01B2  3015
66666666 AAAAAAAA 55555555  01B2  3016 4$:     .LONG   ^X55555555,^XAAAAAAAA,^X66666666 ; LATCH = 30   K = 01-03
0001FFFE 01FE01FE 1E1E1E1E  01BE  3017          .LONG   ^X1E1E1E1E,^X01FE01FE,^X0001FFFE ;               K = 04-06
CCCCCCCC 55555555 AAAAAAAA  01CA  3018          .LONG   ^XAAAAAAAA,^X55555555,^XCCCCCCCC ; LATCH = 29   K = 07-09
0003FFFC 03FC03FC 3C3C3C3C  01D6  3019          .LONG   ^X3C3C3C3C,^X03FC03FC,^X0003FFFC ;               K = 0A-0C
33333333 55555555 AAAAAAAA  01E2  3020          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; LATCH = 27   K = 0D-0F
000FFFF0 0FF00FF0 F0F0F0F0  01EE  3021          .LONG   ^XF0F0F0F0,^X0FF00FF0,^X000FFFF0 ;               K = 10-12
33333333 55555555 AAAAAAAA  01FA  3022          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; LATCH = 23   K = 13-15
00FFFF00 FF00FF00 0F0F0F0F  0206  3023          .LONG   ^X0F0F0F0F,^XFF00FF00,^X00FFFF00 ;               K = 16-18
33333333 55555555 AAAAAAAA  0212  3024          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; LATCH = 15   K = 19-1B
FFFF0000 00FF00FF 0F0F0F0F  021E  3025          .LONG   ^X0F0F0F0F,^X00FF00FF,^XFFFF0000 ;               K = 1C-1E
                        022A  3026
                        022A  3027 ; DSIZE Data for Subtest 3
                        022A  3028
                     00  022A  3029 5$:     .BYTE   0       ; DTYPE = BYTE
                     01  022B  3030          .BYTE   1       ; DTYPE = WORD
                     02  022C  3031          .BYTE   2       ; DTYPE = LONGWORD
                        022D  3032
```

```
                                022D  3033 ; Expected Data for Subtest 3
                                022D  3034 ; The numbers on the right indicate values for DTYPE and the index K.
                                022D  3035
        33333333 55555555 AAAAAAAA  022D  3036 6$:     .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; DTYPE = BYTE K = 01-03
        FF0000FF FF00FF00 0F0F0F0F  0239  3037         .LONG   ^X0F0F0F0F,^XFF00FF00,^XFF0000FF ;                 K = 04-06
        33333333 55555555 AAAAAAAA  0245  3038         .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; DTYPE = WORD K = 07-09
        FFFF0000 00FF00FF 0F0F0F0F  0251  3039         .LONG   ^X0F0F0F0F,^X00FF00FF,^XFFFF0000 ;                 K = 0A-0C
        33333333 55555555 AAAAAAAA  025D  3040         .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; DTYPE = LONG K = 0D-0F
        00FFFF00 FF00FF00 0F0F0F0F  0269  3041         .LONG   ^X0F0F0F0F,^XFF00FF00,^X00FFFF00 ;                 K = 10-12
                                0275  3042
                                0275  3043 ; Microwords to load the PLATCH and the SLATCH.
                                0275  3044
                                0275  3045 10$:
U 01, 7700,1800,EF64,0300,0470,1802  0275  3046 ;% 01   PL_[0]  ; PLATCH GETS THE CONTENTS OF THE LITRL FIELD
                                0280  3050
                                0280  3051 11$:
U 02, 7700,9800,F764,0300,0470,1803  0280  3052 ;% 02   SL_[0]  ; SLATCH GETS THE CONTENTS OF THE LITRL FIELD
                                028B  3056
                                028B  3057 ; Microword to test PLATCH + SLATCH rotation of the RBUS for Subtests 1 and 2
                                028B  3058
                                028B  3059 12$:
U 04, 6700,4800,9B47,0300,0470,1805  028B  3060 ;% 04   RSRC/TEMP0,ROT/RR.RR.PS,ALPCTL/WX_S.Q_0,RDM_WB
                                0296  3064
                                0296  3065 ; Microword to test size rotation of the RBUS for Subtest 3
                                0296  3066
                                0296  3067 13$:
U 04, 6701,0800,2B47,0000,0470,1805  0296  3068 ;% 04   RSRC/TEMP0,DTYPE/BYTE,ROT/RR.RR.SIZ,ALPCTL/WX_S.Q_0,RDM_WB,VSIZE/1
                                02A1  3072
                                02A1  3073 BEGIN_CPU_DATA
                                0002  3074
                                0002  3075 DCS_DATA
                                0001  3076
U 00, 7700,C800,0364,0300,0470,1801  0001  3077 ;% 00   NOP
U 01, 7700,1800,EF64,0300,0470,1802  000C  3081 ;% 01   PL_[0]
U 02, 7700,9800,F764,0300,0470,1803  0017  3085 ;% 02   SL_[0]
U 03, 5700,8840,0364,0300,0470,1804  0022  3089 ;% 03   R[TEMP0]_RDM
U 04, 6700,4800,9B47,0300,0470,1805  002D  3093 ;% 04   RSRC/TEMP0,ROT/RR.RR.PS,ALPCTL/WX_S.Q_0,RDM_WB
U 05, 7300,4800,0364,0300,0470,1806  0038  3097 ;% 05   STOP.CLOCK
                                0043  3101
                                0043  3102 RTEMP_DATA
                                0001  3103
                     00000000   0001  3104         .LONG   0               ; INITIALIZES RTEMP0 AND MTEMP0
                                0005  3105
                                0005  3106 END_CPU_DATA
                                02A1  3107
                                02A1  3108 END_TEST_DATA
                                02A1
                                02A1        ;-----------------------------------------------------------------------
                                02A1  3109
                                02A1  3110 ENDTEST
```

```
0032        .SBTTL  "TEST   04B     Rotator Test 25 Right Rotation of MBUS Test 1
0032  3112 ;*******************************************************************************
0032  3113 ;++
0032  3114 ;
0032  3115 ; FUNCTIONAL DESCRIPTION
0032  3116 ;
0032  3117 ;       This test examines the rotator's ability to do right circular rotation
0032  3118 ;       of the data on the MBUS. This test is implemented in two subtests.
0032  3119 ;       The first subtest examines the second level shifter inside the ALP gate
0032  3120 ;       array. This is done by using PLATCH values of 0,1,2 and 3. The second
0032  3121 ;       subtest examines the first level shifter inside the SRM gate array by
0032  3122 ;       using PLATCH values of 4,8,12,16,20,24 and 28.
0032  3123 ;
0032  3124 ; TEST ALGORITHM
0032  3125 ;
0032  3126 ;       Subtest 1
0032  3127 ;
0032  3128 ;       This subtest examines the second level shifter inside ALP.
0032  3129 ;
0032  3130 ;        1. Initialize the CPU.
0032  3131 ;        2. Load the index K with a 1.
0032  3132 ;        3. Start signature analysis.
0032  3133 ;        4. Prepare a loop on I for 4 PLATCH values.
0032  3134 ;        5. Load the LITRL field of PL_[LITRL] with a PLATCH value.
0032  3135 ;        6. Load the PL_[LITRL] instruction into DCS.
0032  3136 ;        7. Prepare a loop on J for 6 test patterns.
0032  3137 ;        8. Generate a stuck at zero or one test pattern.
0032  3138 ;        9. Load the pattern into WDREG.
0032  3139 ;       10. Loop on error from here.
0032  3140 ;       11. Execute the micro-code to:
0032  3141 ;                1. Load the PLATCH with the LITRL field.
0032  3142 ;                2. Load MTEMP0 with the data in WDREG.
0032  3143 ;                3. Execute a ROT/RR.MM.P on MTEMP0 and put result into WHREG.
0032  3144 ;       12. Compare the data in WHREG with data from a table indexed by I.
0032  3145 ;       13. If the two do not match, signal an error.
0032  3146 ;       14. Increment the index K.
0032  3147 ;       15. Continue the loop for 6 test patterns.
0032  3148 ;       16. Continue the loop for 4 PLATCH values.
0032  3149 ;       17. Stop signature analysis.
0032  3150 ;
0032  3151 ;       Subtest 2
0032  3152 ;
0032  3153 ;       This subtest examines the first level shifter inside SRM.
0032  3154 ;
0032  3155 ;        1. Load the index K with a 1.
0032  3156 ;        2. Start signature analysis.
0032  3157 ;        3. Prepare a loop on I for 7 PLATCH values.
0032  3158 ;        4. Load the LITRL field of PL_[LITRL] with a PLATCH value.
0032  3159 ;        5. Load the PL_[LITRL] instruction into DCS.
0032  3160 ;        6. Prepare a loop on J for 6 test patterns.
0032  3161 ;        7. Generate a stuck at zero or one test pattern.
0032  3162 ;        8. Load the pattern into WDREG.
0032  3163 ;        9. Loop on error from here.
0032  3164 ;       10. Execute the micro-code to:
0032  3165 ;                1. Load the PLATCH with the LITRL field.
```

```
0032  3166 ;                    2. Load MTEMP0 with the data in WDREG.
0032  3167 ;                    3. Execute a ROT/RR.MM.P on MTEMP0 and put result into WHREG.
0032  3168 ;          11. Compare the data in WHREG with data from a table indexed by I.
0032  3169 ;          12. If the two do not match, signal an error.
0032  3170 ;          13. Increment the index K.
0032  3171 ;          14. Continue the loop for 6 test patterns.
0032  3172 ;          15. Continue the loop for 7 PLATCH values.
0032  3173 ;          16. Stop signature analysis.
0032  3174 ;
0032  3175 ; TEST DATA
0032  3176 ;
0032  3177 ;          The data patterns input to the rotator over the MBUS are the standard
0032  3178 ;          stuck at zero or one test pattterns used in most register tests.
0032  3179 ;          The expected data consists of the same data rotated right the proper
0032  3180 ;          number of bits. By using these data patterns it is possible to find any
0032  3181 ;          stuck bit in the rotator data path as well as any bit tied to any other
0032  3182 ;          bit, even after it has undergone a rotation.
0032  3183 ;
0032  3184 ;          Subtest 1
0032  3185 ;
0032  3186 ;          Index I = PLATCH test data being used.
0032  3187 ;          Index J = Stuck at zero or one test pattern being used.
0032  3188 ;          Index K = Index to expected data table.
0032  3189 ;
0032  3190 ;          I  J  K          PLATCH          MBUS            WBUS
0032  3191 ;
0032  3192 ;          01 01 01         00000000        AAAAAAAA        AAAAAAAA
0032  3193 ;          01 02 02         00000000        55555555        55555555
0032  3194 ;          01 03 03         00000000        33333333        33333333
0032  3195 ;          01 04 04         00000000        0F0F0F0F        0F0F0F0F
0032  3196 ;          01 05 05         00000000        00FF00FF        00FF00FF
0032  3197 ;          01 06 06         00000000        0000FFFF        0000FFFF
0032  3198 ;          02 01 07         00000001        AAAAAAAA        55555555
0032  3199 ;          02 02 08         00000001        55555555        AAAAAAAA
0032  3200 ;          02 03 09         00000001        33333333        99999999
0032  3201 ;          02 04 0A         00000001        0F0F0F0F        87878787
0032  3202 ;          02 05 0B         00000001        00FF00FF        807F807F
0032  3203 ;          02 06 0C         00000001        0000FFFF        80007FFF
0032  3204 ;          03 01 0D         00000002        AAAAAAAA        AAAAAAAA
0032  3205 ;          03 02 0E         00000002        55555555        55555555
0032  3206 ;          03 03 0F         00000002        33333333        CCCCCCCC
0032  3207 ;          03 04 10         00000002        0F0F0F0F        C3C3C3C3
0032  3208 ;          03 05 11         00000002        00FF00FF        C03FC03F
0032  3209 ;          03 06 12         00000002        0000FFFF        C0003FFF
0032  3210 ;          04 01 13         00000003        AAAAAAAA        55555555
0032  3211 ;          04 02 14         00000003        55555555        AAAAAAAA
0032  3212 ;          04 03 15         00000003        33333333        66666666
0032  3213 ;          04 04 16         00000003        0F0F0F0F        E1E1E1E1
0032  3214 ;          04 05 16         00000003        00FF00FF        E01FE01F
0032  3215 ;          04 06 17         00000003        0000FFFF        E0001FFF
0032  3216 ;
0032  3217 ;          Subtest 2
0032  3218 ;
0032  3219 ;          Index I = PLATCH test data being used.
0032  3220 ;          Index J = Stuck at zero or one test pattern being used.
```

```
0032  3221 ;       Index K = Index to expected data table.
0032  3222 ;
0032  3223 ;       I  J  K          PLATCH          MBUS            WBUS
0032  3224 ;
0032  3225 ;       01 01 01        00000004        AAAAAAAA        AAAAAAAA
0032  3226 ;       01 02 02        00000004        55555555        55555555
0032  3227 ;       01 03 03        00000004        33333333        33333333
0032  3228 ;       01 04 04        00000004        0F0F0F0F        F0F0F0F0
0032  3229 ;       01 05 05        00000004        00FF00FF        F00FF00F
0032  3230 ;       01 06 06        00000004        0000FFFF        F0000FFF
0032  3231 ;       02 01 07        00000008        AAAAAAAA        AAAAAAAA
0032  3232 ;       02 02 08        00000008        55555555        55555555
0032  3233 ;       02 03 09        00000008        33333333        33333333
0032  3234 ;       02 04 0A        00000008        0F0F0F0F        0F0F0F0F
0032  3235 ;       02 05 0B        00000008        00FF00FF        FF00FF00
0032  3236 ;       02 06 0C        00000008        0000FFFF        FF0000FF
0032  3237 ;       03 01 0D        0000000C        AAAAAAAA        AAAAAAAA
0032  3238 ;       03 02 0E        0000000C        55555555        55555555
0032  3239 ;       03 03 0F        0000000C        33333333        33333333
0032  3240 ;       03 04 10        0000000C        0F0F0F0F        F0F0F0F0
0032  3241 ;       03 05 11        0000000C        00FF00FF        0FF00FF0
0032  3242 ;       03 06 12        0000000C        0000FFFF        FFF0000F
0032  3243 ;       04 01 13        00000010        AAAAAAAA        AAAAAAAA
0032  3244 ;       04 02 14        00000010        55555555        55555555
0032  3245 ;       04 03 15        00000010        33333333        33333333
0032  3246 ;       04 04 16        00000010        0F0F0F0F        0F0F0F0F
0032  3247 ;       04 05 17        00000010        00FF00FF        00FF00FF
0032  3248 ;       04 06 18        00000010        0000FFFF        FFFF0000
0032  3249 ;       05 01 19        00000014        AAAAAAAA        AAAAAAAA
0032  3250 ;       05 02 1A        00000014        55555555        55555555
0032  3251 ;       05 03 1B        00000014        33333333        33333333
0032  3252 ;       05 04 1C        00000014        0F0F0F0F        F0F0F0F0
0032  3253 ;       05 05 1D        00000014        00FF00FF        F00FF00F
0032  3254 ;       05 06 1E        00000014        0000FFFF        0FFFF000
0032  3255 ;       06 01 1F        00000018        AAAAAAAA        AAAAAAAA
0032  3256 ;       06 02 20        00000018        55555555        55555555
0032  3257 ;       06 03 21        00000018        33333333        33333333
0032  3258 ;       06 04 22        00000018        0F0F0F0F        0F0F0F0F
0032  3259 ;       06 05 23        00000018        00FF00FF        FF00FF00
0032  3260 ;       06 06 24        00000018        0000FFFF        00FFFF00
0032  3261 ;       07 01 25        0000001C        AAAAAAAA        AAAAAAAA
0032  3262 ;       07 02 26        0000001C        55555555        55555555
0032  3263 ;       07 03 27        0000001C        33333333        33333333
0032  3264 ;       07 04 28        0000001C        0F0F0F0F        F0F0F0F0
0032  3265 ;       07 05 29        0000001C        00FF00FF        0FF00FF0
0032  3266 ;       07 06 2A        0000001C        0000FFFF        000FFFF0
0032  3267 ;
0032  3268 ; LOGIC DESCRIPTION
0032  3269 ;
0032  3270 ;       The right rotation data path is in the SRM and ALP gate arrays on the
0032  3271 ;       DPM module. The right rotation function is decoded from the ROT field
0032  3272 ;       of the micro-word by the SRK and ALK gate arrays also on DPM.
0032  3273 ;       The decoded ROT field generates the control signals SHF1L, SHF0L, PRI1
0032  3274 ;       and PRI0 that control the SRM and ALP.
0032  3275 ;
```

```
                        3276 ; ASSUMPTIONS
                        3277 ;
                        3278 ;       This test assumes the scratchpad RTEMP0 is working as well as the MBUS
                        3279 ;       SBUS, WBUS and the rotator's PLATCH.
                        3280 ;
                        3281 ; ERROR DESCRIPTION
                        3282 ;
                        3283 ;       On error Subtest 1 will call out the ALP,SRM,SRK and ALK gate arrays.
                        3284 ;       On error Subtest 2 will call out the SRM and SRK gate arrays.
                        3285 ;
                        3286 ;       On error both subtests will type out the following information:
                        3287 ;
                        3288 ;       EXPECTED DATA
                        3289 ;       RECEIVED DATA
                        3290 ;       LOOP COUNT I
                        3291 ;       LOOP COUNT J
                        3292 ;       INDEX VALUE K
                        3293 ;
                        3294 ;--
                        3295 ;*******************************************************************************
                        3296
                        3297         SUBTEST         1
                        3298 ;+
                        3299 ; This subtest examines the MBUS right rotation for PLATCH = 0,1,2,3.
                        3300 ;-
                        3301         INITIALIZE                                  ; INITIALIZE THE CPU
                        3302         LDINDEX         K,1                         ; LOAD THE INDEX K WITH A 1
                        3303         LOOP            I,1,4                       ; LOOP ON I FOR 4 PLATCH VALUES
                        3304         LDFIELD         3$,I,W,10$,31,39            ; LOAD VALUE INTO LITRL FIELD
                        3305         LOADDCS         10$,,1,1                    ; LOAD PL_[LITRL] INTO DCS
                        3306         LOOP            J,1,6                       ; LOOP ON J FOR 6 TEST PATTERNS
                        3307         SA01PAT         J,1$                        ; GENERATE A SA01 TEST PATTERN
                        3308         LOADREG         WDREG,1$,,L                 ; LOAD TEST PATTERN INTO WDREG
                        3309         ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                        3310         FETCH           0                           ; EXECUTE THE ROT/RR.MM.P ORDER
                        3311         BRSTCLK         4                           ; AND WRITE RESULT INTO WHREG
                        3312         CMPREG          WHREG,2$,K,L                ; IS WHREG CORRECT?
                        3313         IFERROR         ,<<ALP><SRM><SRK><ALK>>     ; IF NOT, SIGNAL AN ERROR
                        3314         INCINDEX        K                           ; INCREMENT INDEX K
                        3315         ENDLOOP         J                           ; CONTINUE SA01 PATTERNS
                        3316         ENDLOOP         I                           ; CONTINUE PLATCH VALUES
                        3317
                        3318         SUBTEST         2
                        3319 ;+
                        3320 ; This subtest examines MBUS right rotation for PLATCH = 4,8,12,16,20,24,28.
                        3321 ;-
                        3322         LDINDEX         K,1                         ; INITIALIZE THE INDEX K
                        3323         LOOP            I,1,7                       ; LOOP ON I FOR 7 PLATCH VALUES
                        3324         LDFIELD         5$,I,W,10$,31,39            ; LITRL FIELD GETS PLATCH VALUE
                        3325         LOADDCS         10$,,1,1                    ; LOAD PL_[LITRL] INTO DCS
                        3326         LOOP            J,1,6                       ; LOOP ON J FOR 6 TEST PATTERNS
```

```
                        00CA  3327          SA01PAT         J,1$                    ; GENERATE A SA01 TEST PATTERN
                        00D2  3328          LOADREG         WDREG,1$,,L             ; LOAD TEST PATTERN INTO WDREG
                        00DA  3329          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        00DC  3330          FETCH           0                       ; EXECUTE THE ROT/RR.MM.P ORDER
                        00E1  3331          BRSTCLK         4                       ; AND WRITE RESULT TO WHREG
                        00E5  3332          CMPREG          WHREG,4$,K,L            ; IS WHREG CORRECT?
                        00EE  3333          IFERROR         ,<<SRM><SRK>>           ; IF NOT, SIGNAL AN ERROR
                        00F5  3334          INCINDEX        K                       ; INCREMENT THE INDEX K
                        00F8  3335          ENDLOOP         J                       ; CONTINUE SA01 TEST PATTERNS
                        00FB  3336          ENDLOOP         I                       ; CONTINUE PLATCH VALUES
                        00FE  3337          SKIP                                    ; SKIP TO NEXT TEST
                        0105  3338
                        0105  3339 BEGIN_TEST_DATA
                        0105
                        0105          ;-------------------------------------------------------------------------
                        0105  3340
                   00000000   0105  3341 1$:      .LONG   0       ; LOCATION TO HOLD SA01PAT FOR WDREG
                        0109  3342
                        0109  3343 ; Expected data for Subtest 1
                        0109  3344 ; The numbers to the right indicate PLATCH and K index values.
                        0109  3345
33333333 55555555 AAAAAAAA  0109  3346 2$:      .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 0   K = 01-03
0000FFFF 00FF00FF 0F0F0F0F  0115  3347          .LONG   ^X0F0F0F0F,^X00FF00FF,^X0000FFFF ;               K = 04-06
99999999 AAAAAAAA 55555555  0121  3348          .LONG   ^X55555555,^XAAAAAAAA,^X99999999 ; PLATCH = 1   K = 07-09
80007FFF 807F807F 87878787  012D  3349          .LONG   ^X87878787,^X807F807F,^X80007FFF ;               K = 0A-0C
CCCCCCCC 55555555 AAAAAAAA  0139  3350          .LONG   ^XAAAAAAAA,^X55555555,^XCCCCCCCC ; PLATCH = 2   K = 0D-0F
C0003FFF C03FC03F C3C3C3C3  0145  3351          .LONG   ^XC3C3C3C3,^XC03FC03F,^XC0003FFF ;               K = 10-12
66666666 AAAAAAAA 55555555  0151  3352          .LONG   ^X55555555,^XAAAAAAAA,^X66666666 ; PLATCH = 3   K = 13-15
E0001FFF E01FE01F E1E1E1E1  015D  3353          .LONG   ^XE1E1E1E1,^XE01FE01F,^XE0001FFF ;               K = 16-18
                        0169  3354
                        0169  3355 ; PLATCH values for Subtest 1
                        0169  3356
                        0169  3357 3$:      .WORD   0       ; I = 1
                        016B  3358          .WORD   1       ; I = 2
                        016D  3359          .WORD   2       ; I = 3
                        016F  3360          .WORD   3       ; I = 4
                        0171  3361
                        0171  3362 ; Expected data for Subtest 2
                        0171  3363 ; The numbers to the right indicate PLATCH and K index values.
                        0171  3364
33333333 55555555 AAAAAAAA  0171  3365 4$:      .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 4   K = 01-03
F0000FFF F00FF00F F0F0F0F0  017D  3366          .LONG   ^XF0F0F0F0,^XF00FF00F,^XF0000FFF ;               K = 04-06
33333333 55555555 AAAAAAAA  0189  3367          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 8   K = 07-09
FF0000FF FF00FF00 0F0F0F0F  0195  3368          .LONG   ^X0F0F0F0F,^XFF00FF00,^XFF0000FF ;               K = 0A-0C
33333333 55555555 AAAAAAAA  01A1  3369          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 12  K = 0D-0F
FFF0000F 0FF00FF0 F0F0F0F0  01AD  3370          .LONG   ^XF0F0F0F0,^X0FF00FF0,^XFFF0000F ;               K = 10-12
33333333 55555555 AAAAAAAA  01B9  3371          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 16  K = 13-15
FFFF0000 00FF00FF 0F0F0F0F  01C5  3372          .LONG   ^X0F0F0F0F,^X00FF00FF,^XFFFF0000 ;               K = 16-18
33333333 55555555 AAAAAAAA  01D1  3373          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 20  K = 19-1B
0FFFF000 F00FF00F F0F0F0F0  01DD  3374          .LONG   ^XF0F0F0F0,^XF00FF00F,^X0FFFF000 ;               K = 1C-1E
33333333 55555555 AAAAAAAA  01E9  3375          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 24  K = 1F-21
00FFFF00 FF00FF00 0F0F0F0F  01F5  3376          .LONG   ^X0F0F0F0F,^XFF00FF00,^X00FFFF00 ;               K = 22-24
33333333 55555555 AAAAAAAA  0201  3377          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 28  K = 25-27
000FFFF0 0FF00FF0 F0F0F0F0  020D  3378          .LONG   ^XF0F0F0F0,^X0FF00FF0,^X000FFFF0 ;               K = 28-2A
                        0219  3379
```

```
                                0219  3380 ; PLATCH values for Subtest 2
                                0219  3381
                          0004  0219  3382 5$:     .WORD   4       ; I = 1
                          0008  021B  3383         .WORD   8       ; I = 2
                          000C  021D  3384         .WORD   12      ; I = 3
                          0010  021F  3385         .WORD   16      ; I = 4
                          0014  0221  3386         .WORD   20      ; I = 5
                          0018  0223  3387         .WORD   24      ; I = 6
                          001C  0225  3388         .WORD   28      ; I = 7
                                0227  3389
                                0227  3390 ; Microword to load the PLATCH
                                0227  3391
                                0227  3392 10$:
U 01, 7700,1800,EF64,0300,0470,1802   0227  3393 ;% 01   PL_[0]  ; PLATCH GETS THE CONTENTS OF THE LITRL FIELD
                                0232  3397
                                0232  3398 BEGIN_CPU_DATA
                                0002  3399
                                0002  3400 DCS_DATA
                                0001  3401
U 00, 7700,C800,0364,0300,0470,1801   0001  3402 ;% 00   NOP
U 01, 7700,1800,EF64,0300,0470,1802   000C  3406 ;% 01   PL_[0]
U 02, 5700,4860,0364,0300,0470,1803   0017  3410 ;% 02   M[TEMP0]_RDM
U 03, 6700,8800,1747,0300,0470,1804   0022  3414 ;% 03   MSRC/TEMP0,ROT/RR.MM.P,ALPCTL/WX_S.Q_0,RDM_WB
U 04, 7300,4800,0364,0300,0470,1805   002D  3418 ;% 04   STOP.CLOCK
                                0038  3422
                                0038  3423 RTEMP_DATA
                                0001  3424
                          00000000  0001  3425         .LONG   0       ; INITIALIZES MTEMP0
                                0005  3426
                                0005  3427 END_CPU_DATA
                                0232  3428
                                0232  3429 END_TEST_DATA
                                0232
                                0232      ;-------------------------------------------------------------------------------
                                0232  3430
                                0232  3431 ENDTEST
```

```
0032        .SBTTL  "TEST   04C     Rotator Test 26 Right Rotation of MBUS Test 2
0032  3433 ;*******************************************************************************
0032  3434 ;++
0032  3435 ;
0032  3436 ; FUNCTIONAL DESCRIPTION
0032  3437 ;
0032  3438 ;       This test examines the rotator micro-orders RR.RR.PS and RR.RR.SIZ.
0032  3439 ;       The first micro-order rotates the data on the RBUS right by the number
0032  3440 ;       of bits specified from the sum of the contents of the PLATCH and the
0032  3441 ;       SLATCH. The second micro-order right rotates the data on the RBUS by
0032  3442 ;       the number of bits specified in the DTYPE field of the test micro-
0032  3443 ;       instruction.
0032  3444 ;
0032  3445 ; TEST ALGORITHM
0032  3446 ;
0032  3447 ;       This test is implemented in three subtests. The first subtest floats a
0032  3448 ;       zero through a field of ones in the PLATCH with a constant one loaded
0032  3449 ;       into the SLATCH. It then adds the two together and uses the sum to
0032  3450 ;       right rotate the data on the RBUS by that number of bits. The second
0032  3451 ;       subtest is identical to the first, except that the floating zero is
0032  3452 ;       run through the SLATCH with a constant one loaded into the PLATCH.
0032  3453 ;       The third subtest loads the DTYPE field of the test micro-instruction
0032  3454 ;       with the three data types BYTE, WORD and LONGWORD and then checks that
0032  3455 ;       the data on the RBUS is right rotated the correct number of bytes for
0032  3456 ;       each data type. The test algorithms for the subtests are given below:
0032  3457 ;
0032  3458 ;       Subtests 1 and 2
0032  3459 ;
0032  3460 ;        1. Initialize the CPU. (Subtest 1 only)
0032  3461 ;        2. Load a constant 1 into SLATCH (Subtest 1) or PLATCH (Subtest 2).
0032  3462 ;        3. Load the micro-instruction to test the P + S latch rotation.
0032  3463 ;        4. Load the index K with 1.
0032  3464 ;        5. Start signature analysis.
0032  3465 ;        6. Prepare to loop on I for 5 latch patterns.
0032  3466 ;        7. Load the RSRC field of the test micro-instruction to source TEMP0.
0032  3467 ;        8. Prepare a loop on J for 6 data patterns.
0032  3468 ;        9. Generate a test pattern based on J.
0032  3469 ;       10. Load the test pattern into the WDREG.
0032  3470 ;       11. Loop on error from here.
0032  3471 ;       12. Execute the micro-code to:
0032  3472 ;               1. Load PLATCH with a pattern (Subtest 1) or 1 (Subtest 2).
0032  3473 ;               2. Load the SLATCH with 1 (Subtest 1) or a pattern (Subtest 2).
0032  3474 ;               3. Load MTEMP0 with the test data from WDREG.
0032  3475 ;               4. Execute the test instruction and read the result to WHREG.
0032  3476 ;       13. Compare the WHREG contents with the expected data from a table
0032  3477 ;           indexed by K.
0032  3478 ;       14. If WHREG does not equal the expected data, signal an error.
0032  3479 ;       15. Increment the index K.
0032  3480 ;       16. Loop back to step 8 for rest of the data patterns.
0032  3481 ;       17. Loop back to step 6 for rest of latch patterns.
0032  3482 ;       18. End signature analysis.
0032  3483 ;
0032  3484 ;       Subtest 3
0032  3485 ;
0032  3486 ;        1. Load the index K with 1.
```

```
                        0032  3487 ;        2. Start signature analysis.
                        0032  3488 ;        3. Prepare to loop on I for 3 data types.
                        0032  3489 ;        4. Load a data type (either BYTE, WORD or LONG) into the DTYPE field
                        0032  3490 ;           of the test micro-instruction from the table at 5$ indexed by I.
                        0032  3491 ;        5. Load the test micro-instruction for Subtest 3 into DCS.
                        0032  3492 ;        6. Prepare a loop on J for 6 data patterns.
                        0032  3493 ;        7. Generate a standard stuck at zero or one test pattern based on J.
                        0032  3494 ;        8. Load the test pattern into the WDREG.
                        0032  3495 ;        9. Loop on error from here.
                        0032  3496 ;       10. Execute the micro-code to:
                        0032  3497 ;               1. Load MTEMP0 with the test pattern from WDREG.
                        0032  3498 ;               2. Rotate MTEMP0 right by DTYPE and write the result to WHREG.
                        0032  3499 ;       11. Compare the data in WHREG with the expected data from the table at
                        0032  3500 ;           6$ indexed by K.
                        0032  3501 ;       12. If WHREG does not equal the expected data, signal an error.
                        0032  3502 ;       13. Increment the index K.
                        0032  3503 ;       14. Continue the loop on J for the remaining data patterns.
                        0032  3504 ;       15. Continue the loop on I for the remaining data types.
                        0032  3505 ;
                        0032  3506 ; TEST DATA
                        0032  3507 ;
                        0032  3508 ;       The test data for Subtests 1 and 2 consist of a floating 0 pattern
                        0032  3509 ;       through a field of ones. This gives a good test of the SRK ALU by
                        0032  3510 ;       causing carries to ripple across bits <0:4> of the SLATCH and PLATCH.
                        0032  3511 ;       Bit <5> of the SLATCH and PLATCH is used for sign only and does not
                        0032  3512 ;       affect the number of bits shifted.
                        0032  3513 ;
                        0032  3514 ;       The data for Subtest 3 uses the 3 data types that can be loaded into
                        0032  3515 ;       the DTYPE field of the test micro-instruction; BYTE, WORD and LONG.
                        0032  3516 ;
                        0032  3517 ; LOGIC DESCRIPTION
                        0032  3518 ;
                        0032  3519 ;       The logic used to execute the right rotation of data on the RBUS is
                        0032  3520 ;       internal to the SRM and ALP gate arrays on the DPM module. The ALP
                        0032  3521 ;       gate array contains the second level shifter which can rotate data on
                        0032  3522 ;       the RBUS by 0,1,2 or 3 bits. The SRM contains the first level shifter
                        0032  3523 ;       which can rotate data on the RBUS from 0 to 7 bytes at a time. The
                        0032  3524 ;       SRK and ALK gate arrays are responsible for controlling the actions of
                        0032  3525 ;       this two level rotator.
                        0032  3526 ;
                        0032  3527 ; ASSUMPTIONS
                        0032  3528 ;
                        0032  3529 ;       This test assumes that the scratchpads MTEMP0 and RTEMP1 are working as
                        0032  3530 ;       well as the RBUS, SBUS, WBUS and the rotator's PLATCH and SLATCH.
                        0032  3531 ;
                        0032  3532 ; ERROR DESCRIPTION
                        0032  3533 ;
                        0032  3534 ;       On error Subtests 1 and 2 call out the SRK,SRM,ALP and ALK gate arrays.
                        0032  3535 ;       On error Subtest 3 will call out the SRK and SRM gate arrays.
                        0032  3536 ;
                        0032  3537 ;       On error all subtests will also type out the following data:
                        0032  3538 ;
                        0032  3539 ;       EXPECTED DATA
                        0032  3540 ;       RECEIVED DATA
                        0032  3541 ;       LOOP COUNT I
```

```
                        0032  3542 ;       LOOP COUNT J
                        0032  3543 ;       INDEX VALUE K
                        0032  3544 ;
                        0032  3545 ;--
                        0032  3546 ;*******************************************************************************
                        0032  3547
                        0032  3548         SUBTEST         1
                        0035
                        0035       ;////////////////////////////////////////////////////////////////////////////
                        0035  3549 ;+
                        0035  3550 ; Tests PLATCH + SLATCH rotation with PLATCH = 30,29,27,23,15 SLATCH = 1.
                        0035  3551 ;-
                        0035  3552         INITIALIZE                                  ; INITIALIZE THE CPU
                        0037  3553         LDFIELD         3$,,W,11$,31,39             ; LOAD A 1 INTO THE LITRL FIELD
                        0048  3554         LOADDCS         11$,,2,1                    ; LOAD SL_[1] INTO DCS
                        004F  3555         LOADDCS         12$,,4,1                    ; LOAD ROT/RR.MM.PS INTO DCS
                        0056  3556         LDINDEX         K,1                         ; INITIALIZE THE INDEX K WITH 1
                        005D  3557         LOOP            I,1,5                       ; LOOP ON I FOR 5 PLATCH VALUES
                        0064  3558         LDFIELD         2$,I,W,10$,31,39            ; LOAD LITRL FIELD WITH A VALUE
                        0075  3559         LOADDCS         10$,,1,1                    ; LOAD PL_[VALUE] INTO DCS
                        007C  3560         LOOP            J,1,6                       ; LOOP ON J FOR 6 TEST PATTERNS
                        0083  3561         SA01PAT         J,1$                        ; GENERATE A SA01 TEST PATTERN
                        008B  3562         LOADREG         WDREG,1$,,L                 ; LOAD TEST PATTERN INTO WDREG
                        0093  3563         ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                        0095  3564         FETCH           0                           ; EXECUTE THE ROT/RR.MM.PS AND
                        009A  3565         BRSTCLK         5                           ; WRITE THE RESULT INTO WHREG
                        009E  3566         CMPREG          WHREG,4$,K,L                ; IS WHREG CORRECT?
                        00A7  3567         IFERROR         ,<<SRK><SRM><ALP><ALK>>     ; IF NOT, SIGNAL AN ERROR
                        00B0  3568         INCINDEX        K                           ; INCREMENT THE INDEX K
                        00B3  3569         ENDLOOP         J                           ; CONTINUE THE SA01 PATTERNS
                        00B6  3570         ENDLOOP         I                           ; CONTINUE THE PLATCH VALUES
                        00B9  3571
                        00B9  3572         SUBTEST         2
                        00BC
                        00BC       ;////////////////////////////////////////////////////////////////////////////
                        00BC  3573 ;+
                        00BC  3574 ; Tests PLATCH + SLATCH rotation with SLATCH = 30,29,27,23,15 PLATCH = 1.
                        00BC  3575 ;-
                        00BC  3576         LDFIELD         3$,,W,10$,31,39             ; LOAD A 1 INTO THE LITRL FIELD
                        00CD  3577         LOADDCS         10$,,1,1                    ; LOAD PL_[1] INTO DCS
                        00D4  3578         LDINDEX         K,1                         ; INITIALIZE THE INDEX K WITH 1
                        00DB  3579         LOOP            I,1,5                       ; LOOP ON I FOR 5 SLATCH VALUES
                        00E2  3580         LDFIELD         2$,I,W,11$,31,39            ; LOAD A FLOATING ZERO PATTERN
                        00F3  3581         LOADDCS         11$,,2,1                    ; FROM LITRL FIELD TO SLATCH
                        00FA  3582         LOOP            J,1,6                       ; LOOP ON J FOR 6 TEST PATTERNS
                        0101  3583         SA01PAT         J,1$                        ; GENERATE A SA01 TEST PATTERN
                        0109  3584         LOADREG         WDREG,1$,,L                 ; LOAD SA01 PATTERN INTO WDREG
                        0111  3585         ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                        0113  3586         FETCH           0                           ; EXECUTE THE ROT/RR.MM.PS AND
                        0118  3587         BRSTCLK         5                           ; WRITE THE RESULT INTO WHREG
                        011C  3588         CMPREG          WHREG,4$,K,L                ; IS WHREG CORRECT?
                        0125  3589         IFERROR         ,<<SRK><SRM><ALP><ALK>>     ; IF NOT, SIGNAL AN ERROR
                        012E  3590         INCINDEX        K                           ; INCREMENT INDEX K
                        0131  3591         ENDLOOP         J                           ; CONTINUE FOR 6 TEST PATTERNS
                        0134  3592         ENDLOOP         I                           ; CONTINUE FOR 5 SLATCH VALUES
```

```
                        0137  3593
                        0137  3594          SUBTEST         3
                        013A
                        013A          ;////////////////////////////////////////////////////////////////////////////
                        013A  3595 ;+
                        013A  3596 ; Tests SIZE rotation.
                        013A  3597 ;-
                        013A  3598          LDINDEX         K,1                     ; INITIALIZE THE INDEX K
                        0141  3599          LOOP            I,1,3                   ; LOOP ON I FOR 3 DATA TYPES
                        0148  3600          LDFIELD         5$,I,B,13$,40,41        ; LOAD DTYPE FIELD OF THE TEST
                        0159  3601          LOADDCS         13$,,4,1                ; INST. AND LOAD IT INTO DCS
                        0160  3602          LOOP            J,1,6                   ; LOOP ON J FOR 6 TEST PATTERNS
                        0167  3603          SA01PAT         J,1$                    ; GENERATE A SA01 TEST PATTERN
                        016F  3604          LOADREG         WDREG,1$,,L             ; LOAD TEST PATTERN INTO WDREG
                        0177  3605          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        0179  3606          FETCH           0                       ; EXECUTE ROT/RR.MM.SIZ AND
                        017E  3607          BRSTCLK         5                       ; WRITE RESULT INTO WHREG
                        0182  3608          CMPREG          WHREG,6$,K,L            ; IS WHREG CORRECT?
                        018B  3609          IFERROR         ,<<SRK><SRM>>           ; IF NOT, SIGNAL AN ERROR
                        0192  3610          INCINDEX        K                       ; INCREMENT K FOR NEXT PATTERN
                        0195  3611          ENDLOOP         J                       ; CONTINUE FOR 6 TEST PATTERNS
                        0198  3612          ENDLOOP         I                       ; CONTINUE FOR 3 DATA TYPES
                        019B  3613          SKIP                                    ; SKIP TO NEXT TEST
                        01A2  3614
                        01A2  3615 BEGIN_TEST_DATA
                        01A2
                        01A2          ;-------------------------------------------------------------------------
                        01A2  3616
                00000000  01A2  3617 1$:      .LONG   0       ; LOCATION TO HOLD SA01PAT FOR WDREG
                        01A6  3618
                        01A6  3619 ; SLATCH and PLATCH values for Subtests 1 and 2
                        01A6  3620
      000F 0017 001B 001D 001E  01A6  3621 2$:      .WORD   ^X1E,^X1D,^X1B,^X17,^X0F        ; FLOATING ZERO PATTERN
                        0001  01B0  3622 3$:      .WORD   ^X01                            ; CONSTANT 1
                        01B2  3623
                        01B2  3624 ; Expected data for Subtests 1 and 2
                        01B2  3625 ; The numbers to the right indicate LATCH and K index values.
                        01B2  3626
  66666666 AAAAAAAA 55555555  01B2  3627 4$:      .LONG   ^X55555555,^XAAAAAAAA,^X66666666 ; LATCH = 30   K = 01-03
  0001FFFE 01FE01FE 1E1E1E1E  01BE  3628          .LONG   ^X1E1E1E1E,^X01FE01FE,^X0001FFFE ;              K = 04-06
  CCCCCCCC 55555555 AAAAAAAA  01CA  3629          .LONG   ^XAAAAAAAA,^X55555555,^XCCCCCCCC ; LATCH = 29   K = 07-09
  0003FFFC 03FC03FC 3C3C3C3C  01D6  3630          .LONG   ^X3C3C3C3C,^X03FC03FC,^X0003FFFC ;              K = 0A-0C
  33333333 55555555 AAAAAAAA  01E2  3631          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; LATCH = 27   K = 0D-0E
  000FFFF0 0FF00FF0 F0F0F0F0  01EE  3632          .LONG   ^XF0F0F0F0,^X0FF00FF0,^X000FFFF0 ;              K = 0F-11
  33333333 55555555 AAAAAAAA  01FA  3633          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; LATCH = 23   K = 12-14
  00FFFF00 FF00FF00 0F0F0F0F  0206  3634          .LONG   ^X0F0F0F0F,^XFF00FF00,^X00FFFF00 ;              K = 15-17
  33333333 55555555 AAAAAAAA  0212  3635          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; LATCH = 15   K = 18-1A
  FFFF0000 00FF00FF 0F0F0F0F  021E  3636          .LONG   ^X0F0F0F0F,^X00FF00FF,^XFFFF0000 ;              K = 1B-1D
                        022A  3637
                        022A  3638 ; DSIZE data for Subtest 3
                        022A  3639
                     00 022A  3640 5$:      .BYTE   0       ; I = 1 DTYPE = BYTE
                     01 022B  3641          .BYTE   1       ; I = 2 DTYPE = WORD
                     02 022C  3642          .BYTE   2       ; I = 3 DTYPE = LONGWORD
                        022D  3643
```

```
                                022D  3644 ; Expected data for Subtest 3
                                022D  3645 ; The numbers to the right indicate DTYPE and K index values.
                                022D  3646
        33333333 55555555 AAAAAAAA  022D  3647 6$:     .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; DTYPE = BYTE K = 01-03
        FF0000FF FF00FF00 0F0F0F0F  0239  3648         .LONG   ^X0F0F0F0F,^XFF00FF00,^XFF0000FF ;                 K = 04-06
        33333333 55555555 AAAAAAAA  0245  3649         .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; DTYPE = WORD K = 07-09
        FFFF0000 00FF00FF 0F0F0F0F  0251  3650         .LONG   ^X0F0F0F0F,^X00FF00FF,^XFFFF0000 ;                 K = 0A-0C
        33333333 55555555 AAAAAAAA  025D  3651         .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; DTYPE = LONG K = 0D-0F
        00FFFF00 FF00FF00 0F0F0F0F  0269  3652         .LONG   ^X0F0F0F0F,^XFF00FF00,^X00FFFF00 ;                 K = 10-12
                                0275  3653
                                0275  3654 ; Microwords to load the PLATCH and SLATCH
                                0275  3655
                                0275  3656 10$:
U 01, 7700,1800,EF64,0300,0470,1802  0275  3657 ;% 01   PL_[0]  ; PLATCH GETS CONTENTS OF THE LITRL FIELD
                                0280  3661
                                0280  3662 11$:
U 02, 7700,9800,F764,0300,0470,1803  0280  3663 ;% 02   SL_[0]  ; SLATCH GETS CONTENTS OF THE LITRL FIELD
                                028B  3667
                                028B  3668 ; Microword to test PLATCH + SLATCH rotation
                                028B  3669
                                028B  3670 12$:
U 04, 6700,4800,9747,0300,0470,1805  028B  3671 ;% 04   MSRC/TEMP0,ROT/RR.MM.PS,ALPCTL/WX_S.Q_0,RDM_WB
                                0296  3675
                                0296  3676 ; Microword to test size rotation
                                0296  3677
                                0296  3678 13$:
U 04, 6701,4800,3B47,0000,0470,1805  0296  3679 ;% 04   MSRC/TEMP0,DTYPE/BYTE,ROT/RR.MM.SIZ,ALPCTL/WX_S.Q_0,RDM_WB,VSIZE/1
                                02A1  3683
                                02A1  3684 BEGIN_CPU_DATA
                                0002  3685
                                0002  3686 DCS_DATA
                                0001  3687
U 00, 7700,C800,0364,0300,0470,1801  0001  3688 ;% 00   NOP
U 01, 7700,1800,EF64,0300,0470,1802  000C  3692 ;% 01   PL_[0]
U 02, 7700,9800,F764,0300,0470,1803  0017  3696 ;% 02   SL_[0]
U 03, 5700,C860,0364,0300,0470,1804  0022  3700 ;% 03   M[TEMP0]_RDM
U 04, 6700,4800,9747,0300,0470,1805  002D  3704 ;% 04   MSRC/TEMP0,ROT/RR.MM.PS,ALPCTL/WX_S.Q_0,RDM_WB
U 05, 7300,4800,0364,0300,0470,1806  0038  3708 ;% 05   STOP.CLOCK
                                0043  3712
                                0043  3713 RTEMP_DATA
                                0001  3714
                       00000000 0001  3715         .LONG   0               ; INITIALIZES MTEMP0
                                0005  3716
                                0005  3717 END_CPU_DATA
                                02A1  3718
                                02A1  3719 END_TEST_DATA
                                02A1
                                02A1        ;-------------------------------------------------------------------------
                                02A1  3720
                                02A1  3721 ENDTEST
```

```
                        002F        .SBTTL  "TEST   04D     Rotator Test 27 Left Rotation of RBUS Test
                        002F  3723 ;*******************************************************************************
                        002F  3724 ;++
                        002F  3725 ;
                        002F  3726 ; FUNCTIONAL DESCRIPTION
                        002F  3727 ;
                        002F  3728 ;       This test examines the rotator's ability to do left circular rotation
                        002F  3729 ;       of the data on the RBUS. It is implemented in two subtests. The first
                        002F  3730 ;       subtest examines the second level shifter inside the ALP gate array
                        002F  3731 ;       while the second subtest examines the byte rotator inside the SRM chip.
                        002F  3732 ;
                        002F  3733 ; TEST ALGORITHM
                        002F  3734 ;
                        002F  3735 ;       Subtest 1
                        002F  3736 ;
                        002F  3737 ;       This subtest examines the second level shifter inside the ALP chip.
                        002F  3738 ;
                        002F  3739 ;        1. Load the K index with 1.
                        002F  3740 ;        2. Start signature analysis.
                        002F  3741 ;        3. Set up a loop of 4 iterations on I.
                        002F  3742 ;        4. Prepare the micro-instruction at 10$ to load the PLATCH with a test
                        002F  3743 ;           value from the table at 3$ indexed by I.
                        002F  3744 ;        5. Load the test micro-instruction into DCS.
                        002F  3745 ;        6. Set up a loop of 6 iterations on J.
                        002F  3746 ;        7. Based on the value of J, generate a standard stuck at zero or one
                        002F  3747 ;           test pattern and place it into 1$.
                        002F  3748 ;        8. Load the test pattern from 1$ into WDREG.
                        002F  3749 ;        9. Loop on error from here.
                        002F  3750 ;       10. Execute the micro-code to:
                        002F  3751 ;               1. Load the PLATCH with the value given in step 4.
                        002F  3752 ;               2. Load RTEMP0 with the data in the WDREG.
                        002F  3753 ;               3. Execute a ROT/RL.RR.P micro-order and read the result back
                        002F  3754 ;                  to the WHREG.
                        002F  3755 ;       11. Compare the data in the WHREG with the expected data from the table
                        002F  3756 ;           at 2$ indexed by K.
                        002F  3757 ;       12. If the expected and received data is not the same, issue an error.
                        002F  3758 ;       13. Increment the index K.
                        002F  3759 ;       14. Loop back to step 6 until all six patterns have been tried.
                        002F  3760 ;       15. Loop back to step 3 until all four PLATCH values have been tried.
                        002F  3761 ;       16. Stop signature analysis.
                        002F  3762 ;
                        002F  3763 ;       Subtest 2
                        002F  3764 ;
                        002F  3765 ;       This subtest examines the first level shifter inside the SRM chip.
                        002F  3766 ;
                        002F  3767 ;        1. Load the K index with 1.
                        002F  3768 ;        2. Start signature analysis.
                        002F  3769 ;        3. Set up a loop of 7 iterations on I.
                        002F  3770 ;        4. Prepare the micro-instruction at 10$ to load the PLATCH with a test
                        002F  3771 ;           value from the table at 5$ indexed by I.
                        002F  3772 ;        5. Load the test micro-instruction into DCS.
                        002F  3773 ;        6. Set up a loop of 6 iterations on J.
                        002F  3774 ;        7. Based on the value of J, generate a standard stuck at zero or one
                        002F  3775 ;           test pattern and place it into 1$.
                        002F  3776 ;        8. Load the test pattern from 1$ into WDREG.
```

```
002F  3777 ;        9. Loop on error from here.
002F  3778 ;       10. Execute the micro-code to:
002F  3779 ;               1. Load the PLATCH with the value given in step 4.
002F  3780 ;               2. Load RTEMP0 with the data in the WDREG.
002F  3781 ;               3. Execute a ROT/RL.RR.P micro-order and read the result back
002F  3782 ;                  to the WHREG.
002F  3783 ;       11. Compare the data in the WHREG with the expected data from the table
002F  3784 ;           at 4$ indexed by K.
002F  3785 ;       12. If the expected and received data is not the same, issue an error.
002F  3786 ;       13. Increment the index K.
002F  3787 ;       14. Loop back to step 6 until all 6 patterns have been tried.
002F  3788 ;       15. Loop back to step 3 until all 7 PLATCH values have been tried.
002F  3789 ;       16. Stop signature analysis.
002F  3790 ;
002F  3791 ; TEST DATA
002F  3792 ;
002F  3793 ;       The data patterns input to the rotator over the RBUS are the standard
002F  3794 ;       stuck at zero or one test pattterns used in most register tests.
002F  3795 ;       The expected data consists of the same data rotated left the proper
002F  3796 ;       number of bits. By using these data patterns it is possible to find any
002F  3797 ;       stuck bit in the rotator data path as well as any bit tied to any other
002F  3798 ;       bit, even after it has undergone a rotation.
002F  3799 ;
002F  3800 ;       Subtest 1
002F  3801 ;
002F  3802 ;       Index I = PLATCH test data being used.
002F  3803 ;       Index J = Stuck at zero or one test pattern being used.
002F  3804 ;       Index K = Index to expected data table.
002F  3805 ;
002F  3806 ;       I  J  K          PLATCH          RBUS            WBUS
002F  3807 ;
002F  3808 ;       01 01 01         00000000        AAAAAAAA        AAAAAAAA
002F  3809 ;       01 02 02         00000000        55555555        55555555
002F  3810 ;       01 03 03         00000000        33333333        33333333
002F  3811 ;       01 04 04         00000000        0F0F0F0F        0F0F0F0F
002F  3812 ;       01 05 05         00000000        00FF00FF        00FF00FF
002F  3813 ;       01 06 06         00000000        0000FFFF        0000FFFF
002F  3814 ;       02 01 07         00000001        AAAAAAAA        55555555
002F  3815 ;       02 02 08         00000001        55555555        AAAAAAAA
002F  3816 ;       02 03 09         00000001        33333333        66666666
002F  3817 ;       02 04 0A         00000001        0F0F0F0F        1E1E1E1E
002F  3818 ;       02 05 0B         00000001        00FF00FF        01FE01FE
002F  3819 ;       02 06 0C         00000001        0000FFFF        0001FFFE
002F  3820 ;       03 01 0D         00000002        AAAAAAAA        AAAAAAAA
002F  3821 ;       03 02 0E         00000002        55555555        55555555
002F  3822 ;       03 03 0F         00000002        33333333        CCCCCCCC
002F  3823 ;       03 04 10         00000002        0F0F0F0F        3C3C3C3C
002F  3824 ;       03 05 11         00000002        00FF00FF        03FC03FC
002F  3825 ;       03 06 12         00000002        0000FFFF        0003FFFC
002F  3826 ;       04 01 13         00000003        AAAAAAAA        55555555
002F  3827 ;       04 02 14         00000003        55555555        AAAAAAAA
002F  3828 ;       04 03 15         00000003        33333333        99999999
002F  3829 ;       04 04 16         00000003        0F0F0F0F        78787878
002F  3830 ;       04 05 16         00000003        00FF00FF        07F807F8
002F  3831 ;       04 06 17         00000003        0000FFFF        0007FFF8
```

```
002F  3832 ;
002F  3833 ;       Subtest 2
002F  3834 ;
002F  3835 ;       Index I = PLATCH test data being used.
002F  3836 ;       Index J = Stuck at zero or one test pattern being used.
002F  3837 ;       Index K = Index to expected data table.
002F  3838 ;
002F  3839 ;       I  J  K          PLATCH          RBUS            WBUS
002F  3840 ;
002F  3841 ;       01 01 01        00000004        AAAAAAAA        AAAAAAAA
002F  3842 ;       01 02 02        00000004        55555555        55555555
002F  3843 ;       01 03 03        00000004        33333333        33333333
002F  3844 ;       01 04 04        00000004        0F0F0F0F        F0F0F0F0
002F  3845 ;       01 05 05        00000004        00FF00FF        0FF00FF0
002F  3846 ;       01 06 06        00000004        0000FFFF        000FFFF0
002F  3847 ;       02 01 07        00000008        AAAAAAAA        AAAAAAAA
002F  3848 ;       02 02 08        00000008        55555555        55555555
002F  3849 ;       02 03 09        00000008        33333333        33333333
002F  3850 ;       02 04 0A        00000008        0F0F0F0F        0F0F0F0F
002F  3851 ;       02 05 0B        00000008        00FF00FF        FF00FF00
002F  3852 ;       02 06 0C        00000008        0000FFFF        00FFFF00
002F  3853 ;       03 01 0D        0000000C        AAAAAAAA        AAAAAAAA
002F  3854 ;       03 02 0E        0000000C        55555555        55555555
002F  3855 ;       03 03 0F        0000000C        33333333        33333333
002F  3856 ;       03 04 10        0000000C        0F0F0F0F        F0F0F0F0
002F  3857 ;       03 05 11        0000000C        00FF00FF        F00FF00F
002F  3858 ;       03 06 12        0000000C        0000FFFF        0FFFF000
002F  3859 ;       04 01 13        00000010        AAAAAAAA        AAAAAAAA
002F  3860 ;       04 02 14        00000010        55555555        55555555
002F  3861 ;       04 03 15        00000010        33333333        33333333
002F  3862 ;       04 04 16        00000010        0F0F0F0F        0F0F0F0F
002F  3863 ;       04 05 17        00000010        00FF00FF        00FF00FF
002F  3864 ;       04 06 18        00000010        0000FFFF        FFFF0000
002F  3865 ;       05 01 19        00000014        AAAAAAAA        AAAAAAAA
002F  3866 ;       05 02 1A        00000014        55555555        55555555
002F  3867 ;       05 03 1B        00000014        33333333        33333333
002F  3868 ;       05 04 1C        00000014        0F0F0F0F        F0F0F0F0
002F  3869 ;       05 05 1D        00000014        00FF00FF        0FF00FF0
002F  3870 ;       05 06 1E        00000014        0000FFFF        FFF0000F
002F  3871 ;       06 01 1F        00000018        AAAAAAAA        AAAAAAAA
002F  3872 ;       06 02 20        00000018        55555555        55555555
002F  3873 ;       06 03 21        00000018        33333333        33333333
002F  3874 ;       06 04 22        00000018        0F0F0F0F        0F0F0F0F
002F  3875 ;       06 05 23        00000018        00FF00FF        FF00FF00
002F  3876 ;       06 06 24        00000018        0000FFFF        FF0000FF
002F  3877 ;       07 01 25        0000001C        AAAAAAAA        AAAAAAAA
002F  3878 ;       07 02 26        0000001C        55555555        55555555
002F  3879 ;       07 03 27        0000001C        33333333        33333333
002F  3880 ;       07 04 28        0000001C        0F0F0F0F        F0F0F0F0
002F  3881 ;       07 05 29        0000001C        00FF00FF        F00FF00F
002F  3882 ;       07 06 2A        0000001C        0000FFFF        F0000FFF
002F  3883 ;
002F  3884 ; LOGIC DESCRIPTION
002F  3885 ;
002F  3886 ;       The left rotation data path is in the SRM and ALP gate arrays on the
```

```
                        002F  3887 ;       DPM module. The left rotation function is decoded from the ROT field
                        002F  3888 ;       of the micro-word by the SRK and ALK gate arrays also on DPM.
                        002F  3889 ;       The decoded ROT field generates the control signals SHF1L, SHF0L, PRI1
                        002F  3890 ;       and PRI0 that control the SRM and ALP.
                        002F  3891 ;
                        002F  3892 ; ASSUMPTIONS
                        002F  3893 ;
                        002F  3894 ;       This test assumes the scratchpad RTEMP0 is working as well as the RBUS
                        002F  3895 ;       SBUS, WBUS and the rotator's PLATCH.
                        002F  3896 ;
                        002F  3897 ; ERROR DESCRIPTION
                        002F  3898 ;
                        002F  3899 ;       On error Subtest 1 will call out the ALP,SRM,SRK and ALK gate arrays.
                        002F  3900 ;       On error Subtest 2 will call out the SRM and SRK gate arrays.
                        002F  3901 ;
                        002F  3902 ;       On error both subtests will type out the following information:
                        002F  3903 ;
                        002F  3904 ;       EXPECTED DATA
                        002F  3905 ;       RECEIVED DATA
                        002F  3906 ;       LOOP COUNT I
                        002F  3907 ;       LOOP COUNT J
                        002F  3908 ;       INDEX VALUE K
                        002F  3909 ;
                        002F  3910 ;--
                        002F  3911 ;*****************************************************************************
                        002F  3912
                        002F  3913         SUBTEST         1
                        0032
                        0032       ;////////////////////////////////////////////////////////////////////////////
                        0032  3914 ;*
                        0032  3915 ; This subtest examines the second level shifter inside the ALP gate array
                        0032  3916 ; for PLATCH = 0,1,2,3.
                        0032  3917 ;-
                        0032  3918         INITIALIZE                                  ; INITIALIZE THE CPU
                        0034  3919         LDINDEX         K,1                         ; INITIALIZE THE INDEX K WITH 1
                        003B  3920         LOOP            I,1,4                       ; LOOP ON I FOR 4 PLATCH VALUES
                        0042  3921         LDFIELD         3$,I,W,10$,31,39            ; LOAD LITRL WITH PLATCH VALUE
                        0053  3922         LOADDCS         10$,,1,1                    ; LOAD PL_[LITRL] INST. TO DCS
                        005A  3923         LOOP            J,1,6                       ; LOOP ON J FOR 6 TEST PATTERNS
                        0061  3924         SA01PAT         J,1$                        ; GENERATE A SA01 TEST PATTERN
                        0069  3925         LOADREG         WDREG,1$,,L                 ; LOAD TEST PATTERN INTO WDREG
                        0071  3926         ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                        0073  3927         FETCH           0                           ; EXECUTE ROT/RL.RR.P AND WRITE
                        0078  3928         BRSTCLK         4                           ; THE RESULT INTO WHREG
                        007C  3929         CMPREG          WHREG,2$,K,L                ; IS WHREG CORRECT?
                        0085  3930         IFERROR         ,<<ALP><SRM><SRK><ALK>>     ; IF NOT, SIGNAL AN ERROR
                        008E  3931         INCINDEX        K                           ; INCREMENT K FOR NEXT PATTERN
                        0091  3932         ENDLOOP         J                           ; CONTINUE FOR 6 TEST PATTERNS
                        0094  3933         ENDLOOP         I                           ; CONTINUE FOR 4 PLATCH VALUES
                        0097  3934
                        0097  3935         SUBTEST         2
                        009A
                        009A       ;////////////////////////////////////////////////////////////////////////////
                        009A  3936 ;*
                        009A  3937 ; This subtest examines the first level shifter inside the SRM gate array for
```

```
                                009A  3938 ; PLATCH values of 4,8,12,16,20,24,28.
                                009A  3939 ;-
                                009A  3940          LDINDEX         K,1                             ; INITIALIZE THE INDEX K
                                00A1  3941          LOOP            I,1,7                           ; LOOP ON I FOR 7 PLATCH VALUES
                                00A8  3942          LDFIELD         5$,I,W,10$,31,39                ; LOAD LITRL WITH PLATCH VALUE
                                00B9  3943          LOADDCS         10$,,1,1                        ; LOAD PL_[LITRL] INST. TO DCS
                                00C0  3944          LOOP            J,1,6                           ; LOOP ON J FOR 6 TEST PATTERNS
                                00C7  3945          SA01PAT         J,1$                            ; GENERATE A SA01 TEST PATTERN
                                00CF  3946          LOADREG         WDREG,1$,,L                     ; LOAD TEST PATTERN INTO WDREG
                                00D7  3947          ERRLOOP                                         ; LOOP ON ERROR FROM HERE
                                00D9  3948          FETCH           0                               ; EXECUTE ROT/RL.RR.P ORDER AND
                                00DE  3949          BRSTCLK         4                               ; WRITE RESULT INTO WHREG
                                00E2  3950          CMPREG          WHREG,4$,K,L                    ; IS THE WHREG CORRECT?
                                00EB  3951          IFERROR         ,<<SRM><SRK>>                   ; IF NOT, SIGNAL AN ERROR
                                00F2  3952          INCINDEX        K                               ; INCREMENT K FOR NEXT PATTERN
                                00F5  3953          ENDLOOP         J                               ; CONTINUE FOR 6 TEST PATTERNS
                                00F8  3954          ENDLOOP         I                               ; CONTINUE FOR 7 PLATCH VALUES
                                00FB  3955          SKIP                                            ; SKIP TO NEXT TEST
                                0102  3956
                                0102  3957 BEGIN_TEST_DATA
                                0102
                                0102       ;-----------------------------------------------------------------------
                                0102  3958
                     00000000   0102  3959 1$:      .LONG   0       ; LOCATION TO HOLD SA01PAT FOR WDREG
                                0106  3960
                                0106  3961 ; Expected data for Subtest 1
                                0106  3962 ; The numbers to the right indicate PLATCH and K index values.
                                0106  3963
 33333333 55555555 AAAAAAAA     0106  3964 2$:      .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 0   K = 01-03
 0000FFFF 00FF00FF 0F0F0F0F     0112  3965          .LONG   ^X0F0F0F0F,^X00FF00FF,^X0000FFFF ;               K = 04-06
 66666666 AAAAAAAA 55555555     011E  3966          .LONG   ^X55555555,^XAAAAAAAA,^X66666666 ; PLATCH = 1   K = 07-09
 0001FFFE 01FE01FE 1E1E1E1E     012A  3967          .LONG   ^X1E1E1E1E,^X01FE01FE,^X0001FFFE ;               K = 0A-0C
 CCCCCCCC 55555555 AAAAAAAA     0136  3968          .LONG   ^XAAAAAAAA,^X55555555,^XCCCCCCCC ; PLATCH = 2   K = 0D-0F
 0003FFFC 03FC03FC 3C3C3C3C     0142  3969          .LONG   ^X3C3C3C3C,^X03FC03FC,^X0003FFFC ;               K = 10-12
 99999999 AAAAAAAA 55555555     014E  3970          .LONG   ^X55555555,^XAAAAAAAA,^X99999999 ; PLATCH = 3   K = 13-15
 0007FFF8 07F807F8 78787878     015A  3971          .LONG   ^X78787878,^X07F807F8,^X0007FFF8 ;               K = 16-18
                                0166  3972
                                0166  3973 ; PLATCH values for Subtest 1
                                0166  3974
                          0000  0166  3975 3$:      .WORD   0       ; I = 1
                          0001  0168  3976          .WORD   1       ; I = 2
                          0002  016A  3977          .WORD   2       ; I = 3
                          0003  016C  3978          .WORD   3       ; I = 4
                                016E  3979
                                016E  3980 ; Expected data for Subtest 2
                                016E  3981 ; The numbers to the right indicate PLATCH and K index values.
                                016E  3982
 33333333 55555555 AAAAAAAA     016E  3983 4$:      .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 4   K = 01-03
 000FFFF0 0FF00FF0 F0F0F0F0     017A  3984          .LONG   ^XF0F0F0F0,^X0FF00FF0,^X000FFFF0 ;               K = 04-06
 33333333 55555555 AAAAAAAA     0186  3985          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 8   K = 07-09
 00FFFF00 FF00FF00 0F0F0F0F     0192  3986          .LONG   ^X0F0F0F0F,^XFF00FF00,^X00FFFF00 ;               K = 0A-0C
 33333333 55555555 AAAAAAAA     019E  3987          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 12  K = 0D-0F
 0FFFF000 F00FF00F F0F0F0F0     01AA  3988          .LONG   ^XF0F0F0F0,^XF00FF00F,^X0FFFF000 ;               K = 10-12
 33333333 55555555 AAAAAAAA     01B6  3989          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 16  K = 13-15
 FFFF0000 00FF00FF 0F0F0F0F     01C2  3990          .LONG   ^X0F0F0F0F,^X00FF00FF,^XFFFF0000 ;               K = 16-18
```

```
         33333333 55555555 AAAAAAAA  01CE  3991          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 20  K = 19-1B
         FFF0000F 0FF00FF0 F0F0F0F0  01DA  3992          .LONG   ^XF0F0F0F0,^X0FF00FF0,^XFFF0000F ;                K = 1C-1E
         33333333 55555555 AAAAAAAA  01E6  3993          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 24  K = 1F-21
         FF0000FF FF00FF00 0F0F0F0F  01F2  3994          .LONG   ^X0F0F0F0F,^XFF00FF00,^XFF0000FF ;                K = 22-24
         33333333 55555555 AAAAAAAA  01FE  3995          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 28  K = 25-27
         F0000FFF F00FF00F F0F0F0F0  020A  3996          .LONG   ^XF0F0F0F0,^XF00FF00F,^XF0000FFF ;                K = 28-2A
                                    0216  3997
                                    0216  3998 ; PLATCH values for Subtest 2
                                    0216  3999
                               0004 0216  4000 5$:      .WORD   4       ; I = 1
                               0008 0218  4001          .WORD   8       ; I = 2
                               000C 021A  4002          .WORD   12      ; I = 3
                               0010 021C  4003          .WORD   16      ; I = 4
                               0014 021E  4004          .WORD   20      ; I = 5
                               0018 0220  4005          .WORD   24      ; I = 6
                               001C 0222  4006          .WORD   28      ; I = 7
                                    0224  4007
                                    0224  4008 ; Microword to load the PLATCH
                                    0224  4009
                                    0224  4010 10$:
U 01, 7700,1800,EF64,0300,0470,1802 0224  4011 ;% 01   PL_[0]  ; PLATCH GETS THE CONTENTS OF THE LITRL FIELD
                                    022F  4015
                                    022F  4016 BEGIN_CPU_DATA
                                    0002  4017
                                    0002  4018 DCS_DATA
                                    0001  4019
U 00, 7700,C800,0364,0300,0470,1801 0001  4020 ;% 00   NOP
U 01, 7700,1800,EF64,0300,0470,1802 000C  4024 ;% 01   PL_[0]
U 02, 5700,0840,0364,0300,0470,1803 0017  4028 ;% 02   R[TEMP0]_RDM
U 03, 6700,8800,8B47,0300,0470,1804 0022  4032 ;% 03   RSRC/TEMP0,ROT/RL.RR.P,ALPCTL/WX_S.Q_0,RDM_WB
U 04, 7300,4800,0364,0300,0470,1805 002D  4036 ;% 04   STOP.CLOCK
                                    0038  4040
                                    0038  4041 RTEMP_DATA
                                    0001  4042
                           00000000 0001  4043          .LONG   0       ; INITIALIZES RTEMP0
                                    0005  4044
                                    0005  4045 END_CPU_DATA
                                    022F  4046
                                    022F  4047 END_TEST_DATA
                                    022F
                                    022F        ;-----------------------------------------------------------------------------
                                    022F  4048
                                    022F  4049 ENDTEST
```

```
                        002F        .SBTTL  "TEST   04E     Rotator Test 28 Left Rotation of MBUS Test
                        002F  4051 ;*******************************************************************************
                        002F  4052 ;++
                        002F  4053 ;
                        002F  4054 ; FUNCTIONAL DESCRIPTION
                        002F  4055 ;
                        002F  4056 ;       This test examines the rotator's ability to do left circular rotation
                        002F  4057 ;       of the data on the RBUS. It is implemented in three subtests. The first
                        002F  4058 ;       subtest examines the second level shifter inside the ALP gate array.
                        002F  4059 ;       The second subtest examines the byte rotator inside the SRM chip. The
                        002F  4060 ;       third subtest examines the ROT/RL.MM.SIZ micro-order.
                        002F  4061 ;
                        002F  4062 ; TEST ALGORITHM
                        002F  4063 ;
                        002F  4064 ;       Subtest 1
                        002F  4065 ;
                        002F  4066 ;       This subtest examines the second level shifter inside the ALP chip.
                        002F  4067 ;
                        002F  4068 ;        1. Load the K index with 1.
                        002F  4069 ;        2. Start signature analysis.
                        002F  4070 ;        3. Set up a loop of 4 iterations on I.
                        002F  4071 ;        4. Prepare the micro-instruction at 10$ to load the PLATCH with a test
                        002F  4072 ;           value from the table at 3$ indexed by I.
                        002F  4073 ;        5. Load the test micro-instruction into DCS.
                        002F  4074 ;        6. Set up a loop of 6 iterations on J.
                        002F  4075 ;        7. Based on the value of J, generate a standard stuck at zero or one
                        002F  4076 ;           test pattern and place it into 1$.
                        002F  4077 ;        8. Load the test pattern from 1$ into WDREG.
                        002F  4078 ;        9. Loop on error from here.
                        002F  4079 ;       10. Execute the micro-code to:
                        002F  4080 ;               1. Load the PLATCH with the value given in step 4.
                        002F  4081 ;               2. Load RTEMP0 with the data in the WDREG.
                        002F  4082 ;               3. Execute a ROT/RL.MM.P micro-order and read the result back
                        002F  4083 ;                  to the WHREG.
                        002F  4084 ;       11. Compare the data in the WHREG with the expected data from the table
                        002F  4085 ;           at 2$ indexed by K.
                        002F  4086 ;       12. If the expected and received data is not the same, issue an error.
                        002F  4087 ;       13. Increment the index K.
                        002F  4088 ;       14. Loop back to step 6 until all six patterns have been tried.
                        002F  4089 ;       15. Loop back to step 3 until all four PLATCH values have been tried.
                        002F  4090 ;       16. Stop signature analysis.
                        002F  4091 ;
                        002F  4092 ;       Subtest 2
                        002F  4093 ;
                        002F  4094 ;       This subtest examines the first level shifter inside the SRM chip.
                        002F  4095 ;
                        002F  4096 ;        1. Load the K index with 1.
                        002F  4097 ;        2. Start signature analysis.
                        002F  4098 ;        3. Set up a loop of 7 iterations on I.
                        002F  4099 ;        4. Prepare the micro-instruction at 10$ to load the PLATCH with a test
                        002F  4100 ;           value from the table at 5$ indexed by I.
                        002F  4101 ;        5. Load the test micro-instruction into DCS.
                        002F  4102 ;        6. Set up a loop of 6 iterations on J.
                        002F  4103 ;        7. Based on the value of J, generate a standard stuck at zero or one
                        002F  4104 ;           test pattern and place it into 1$.
```

```
                        002F  4105 ;        8. Load the test pattern from 1$ into WDREG.
                        002F  4106 ;        9. Loop on error from here.
                        002F  4107 ;       10. Execute the micro-code to:
                        002F  4108 ;               1. Load the PLATCH with the value given in step 4.
                        002F  4109 ;               2. Load RTEMP0 with the data in the WDREG.
                        002F  4110 ;               3. Execute a ROT/RL.MM.P micro-order and read the result back
                        002F  4111 ;                  to the WHREG.
                        002F  4112 ;       11. Compare the data in the WHREG with the expected data from the table
                        002F  4113 ;           at 4$ indexed by K.
                        002F  4114 ;       12. If the expected and received data is not the same, issue an error.
                        002F  4115 ;       13. Increment the index K.
                        002F  4116 ;       14. Loop back to step 6 until all 6 patterns have been tried.
                        002F  4117 ;       15. Loop back to step 3 until all 7 PLATCH values have been tried.
                        002F  4118 ;       16. Stop signature analysis.
                        002F  4119 ;
                        002F  4120 ; TEST DATA
                        002F  4121 ;
                        002F  4122 ;       The data patterns input to the rotator over the RBUS are the standard
                        002F  4123 ;       stuck at zero or one test pattterns used in most register tests.
                        002F  4124 ;       The expected data consists of the same data rotated left the proper
                        002F  4125 ;       number of bits. By using these data patterns it is possible to find any
                        002F  4126 ;       stuck bit in the rotator data path as well as any bit tied to any other
                        002F  4127 ;       bit, even after it has undergone a rotation.
                        002F  4128 ;
                        002F  4129 ;       Subtest 1
                        002F  4130 ;
                        002F  4131 ;       Index I = PLATCH test data being used.
                        002F  4132 ;       Index J = Stuck at zero or one test pattern being used.
                        002F  4133 ;       Index K = Index to expected data table.
                        002F  4134 ;
                        002F  4135 ;       I  J  K          PLATCH          RBUS            WBUS
                        002F  4136 ;
                        002F  4137 ;       01 01 01         00000000        AAAAAAAA        AAAAAAAA
                        002F  4138 ;       01 02 02         00000000        55555555        55555555
                        002F  4139 ;       01 03 03         00000000        33333333        33333333
                        002F  4140 ;       01 04 04         00000000        0F0F0F0F        0F0F0F0F
                        002F  4141 ;       01 05 05         00000000        00FF00FF        00FF00FF
                        002F  4142 ;       01 06 06         00000000        0000FFFF        0000FFFF
                        002F  4143 ;       02 01 07         00000001        AAAAAAAA        55555555
                        002F  4144 ;       02 02 08         00000001        55555555        AAAAAAAA
                        002F  4145 ;       02 03 09         00000001        33333333        66666666
                        002F  4146 ;       02 04 0A         00000001        0F0F0F0F        1E1E1E1E
                        002F  4147 ;       02 05 0B         00000001        00FF00FF        01FE01FE
                        002F  4148 ;       02 06 0C         00000001        0000FFFF        0001FFFE
                        002F  4149 ;       03 01 0D         00000002        AAAAAAAA        AAAAAAAA
                        002F  4150 ;       03 02 0E         00000002        55555555        55555555
                        002F  4151 ;       03 03 0F         00000002        33333333        CCCCCCCC
                        002F  4152 ;       03 04 10         00000002        0F0F0F0F        3C3C3C3C
                        002F  4153 ;       03 05 11         00000002        00FF00FF        03FC03FC
                        002F  4154 ;       03 06 12         00000002        0000FFFF        0003FFFC
                        002F  4155 ;       04 01 13         00000003        AAAAAAAA        55555555
                        002F  4156 ;       04 02 14         00000003        55555555        AAAAAAAA
                        002F  4157 ;       04 03 15         00000003        33333333        99999999
                        002F  4158 ;       04 04 16         00000003        0F0F0F0F        78787878
                        002F  4159 ;       04 05 16         00000003        00FF00FF        07F807F8
```

```
002F  4160 ;       04 06 17        00000003        0000FFFF        0007FFF8
002F  4161 ;
002F  4162 ;       Subtest 2
002F  4163 ;
002F  4164 ;       Index I = PLATCH test data being used.
002F  4165 ;       Index J = Stuck at zero or one test pattern being used.
002F  4166 ;       Index K = Index to expected data table.
002F  4167 ;
002F  4168 ;       I  J  K          PLATCH          RBUS            WBUS
002F  4169 ;
002F  4170 ;       01 01 01        00000004        AAAAAAAA        AAAAAAAA
002F  4171 ;       01 02 02        00000004        55555555        55555555
002F  4172 ;       01 03 03        00000004        33333333        33333333
002F  4173 ;       01 04 04        00000004        0F0F0F0F        F0F0F0F0
002F  4174 ;       01 05 05        00000004        00FF00FF        0FF00FF0
002F  4175 ;       01 06 06        00000004        0000FFFF        000FFFF0
002F  4176 ;       02 01 07        00000008        AAAAAAAA        AAAAAAAA
002F  4177 ;       02 02 08        00000008        55555555        55555555
002F  4178 ;       02 03 09        00000008        33333333        33333333
002F  4179 ;       02 04 0A        00000008        0F0F0F0F        0F0F0F0F
002F  4180 ;       02 05 0B        00000008        00FF00FF        FF00FF00
002F  4181 ;       02 06 0C        00000008        0000FFFF        00FFFF00
002F  4182 ;       03 01 0D        0000000C        AAAAAAAA        AAAAAAAA
002F  4183 ;       03 02 0E        0000000C        55555555        55555555
002F  4184 ;       03 03 0F        0000000C        33333333        33333333
002F  4185 ;       03 04 10        0000000C        0F0F0F0F        F0F0F0F0
002F  4186 ;       03 05 11        0000000C        00FF00FF        F00FF00F
002F  4187 ;       03 06 12        0000000C        0000FFFF        0FFFF000
002F  4188 ;       04 01 13        00000010        AAAAAAAA        AAAAAAAA
002F  4189 ;       04 02 14        00000010        55555555        55555555
002F  4190 ;       04 03 15        00000010        33333333        33333333
002F  4191 ;       04 04 16        00000010        0F0F0F0F        0F0F0F0F
002F  4192 ;       04 05 17        00000010        00FF00FF        00FF00FF
002F  4193 ;       04 06 18        00000010        0000FFFF        FFFF0000
002F  4194 ;       05 01 19        00000014        AAAAAAAA        AAAAAAAA
002F  4195 ;       05 02 1A        00000014        55555555        55555555
002F  4196 ;       05 03 1B        00000014        33333333        33333333
002F  4197 ;       05 04 1C        00000014        0F0F0F0F        F0F0F0F0
002F  4198 ;       05 05 1D        00000014        00FF00FF        0FF00FF0
002F  4199 ;       05 06 1E        00000014        0000FFFF        FFF0000F
002F  4200 ;       06 01 1F        00000018        AAAAAAAA        AAAAAAAA
002F  4201 ;       06 02 20        00000018        55555555        55555555
002F  4202 ;       06 03 21        00000018        33333333        33333333
002F  4203 ;       06 04 22        00000018        0F0F0F0F        0F0F0F0F
002F  4204 ;       06 05 23        00000018        00FF00FF        FF00FF00
002F  4205 ;       06 06 24        00000018        0000FFFF        FF0000FF
002F  4206 ;       07 01 25        0000001C        AAAAAAAA        AAAAAAAA
002F  4207 ;       07 02 26        0000001C        55555555        55555555
002F  4208 ;       07 03 27        0000001C        33333333        33333333
002F  4209 ;       07 04 28        0000001C        0F0F0F0F        F0F0F0F0
002F  4210 ;       07 05 29        0000001C        00FF00FF        F00FF00F
002F  4211 ;       07 06 2A        0000001C        0000FFFF        F0000FFF
002F  4212 ;
002F  4213 ; LOGIC DESCRIPTION
002F  4214 ;
```

```
                        002F  4215 ;       The left rotation data path is in the SRM and ALP gate arrays on the
                        002F  4216 ;       DPM module. The left rotation function is decoded from the ROT field
                        002F  4217 ;       of the micro-word by the SRK and ALK gate arrays also on DPM.
                        002F  4218 ;       The decoded ROT field generates the control signals SHF1L, SHF0L, PRI1
                        002F  4219 ;       and PRI0 that control the SRM and ALP.
                        002F  4220 ;
                        002F  4221 ; ASSUMPTIONS
                        002F  4222 ;
                        002F  4223 ;       This test assumes the scratchpad RTEMP0 is working as well as the RBUS
                        002F  4224 ;       SBUS, WBUS and the rotator's PLATCH.
                        002F  4225 ;
                        002F  4226 ; ERROR DESCRIPTION
                        002F  4227 ;
                        002F  4228 ;       On error Subtest 1 will call out the ALP,SRM,SRK and ALK gate arrays.
                        002F  4229 ;       On error Subtest 2 will call out the SRM and SRK gate arrays.
                        002F  4230 ;
                        002F  4231 ;       On error both subtests will type out the following information:
                        002F  4232 ;
                        002F  4233 ;       EXPECTED DATA
                        002F  4234 ;       RECEIVED DATA
                        002F  4235 ;       LOOP COUNT I
                        002F  4236 ;       LOOP COUNT J
                        002F  4237 ;       INDEX VALUE K
                        002F  4238 ;
                        002F  4239 ;--
                        002F  4240 ;*****************************************************************************
                        002F  4241
                        002F  4242         SUBTEST          1
                        0032
                        0032           ;/////////////////////////////////////////////////////////////////////////
                        0032  4243 ;*
                        0032  4244 ; This subtest examines the first level shifter on the ALP gate array.
                        0032  4245 ;-
                        0032  4246         INITIALIZE                                  ; INITIALIZE THE CPU
                        0034  4247         LDINDEX          K,1                        ; INITIALIZE THE INDEX K WITH 1
                        003B  4248         LOADDCS          11$,,3,1                   ; LOAD ROT/RL.MM.P INSTRUCTION
                        0042  4249         LOOP             I,1,4                      ; LOOP ON I FOR 4 PLATCH VALUES
                        0049  4250         LDFIELD          3$,I,W,10$,31,39           ; LOAD LITRL FIELD WITH VALUE
                        005A  4251         LOADDCS          10$,,1,1                   ; LOAD PL_[LITRL] INTO DCS
                        0061  4252         LOOP             J,1,6                      ; LOOP ON J FOR 6 TEST PATTERNS
                        0068  4253         SA01PAT          J,1$                       ; GENERATE A SA01 TEST PATTERN
                        0070  4254         LOADREG          WDREG,1$,,L                ; LOAD THE PATTERN INTO WDREG
                        0078  4255         ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                        007A  4256         FETCH            0                          ; EXECUTE THE ROT/RL.MM.P ORDER
                        007F  4257         BRSTCLK          4                          ; AND WRITE THE RESULT TO WHREG
                        0083  4258         CMPREG           WHREG,2$,K,L               ; IS THE WHREG CORRECT?
                        008C  4259         IFERROR          ,<<ALP><SRM><SRK><ALK>>    ; IF NOT, SIGNAL AN ERROR
                        0095  4260         INCINDEX         K                          ; INCREMENT K FOR NEXT PATTERN
                        0098  4261         ENDLOOP          J                          ; CONTINUE FOR 6 TEST PATTERNS
                        009B  4262         ENDLOOP          I                          ; CONTINUE FOR 4 PLATCH VALUES
                        009E  4263
                        009E  4264         SUBTEST          2
                        00A1
                        00A1           ;/////////////////////////////////////////////////////////////////////////
                        00A1  4265 ;*
```

```
                        00A1  4266 ; This subtest examines the first level shifter inside the SRM gate array.
                        00A1  4267 ;-
                        00A1  4268          LDINDEX         K,1                         ; INITIALIZE THE INDEX K WITH 1
                        00A8  4269          LOOP            I,1,7                       ; LOOP ON I FOR 7 PLATCH VALUES
                        00AF  4270          LDFIELD         5$,I,W,10$,31,39            ; LOAD LITRL FIELD WITH A VALUE
                        00C0  4271          LOADDCS         10$,,1,1                    ; LOAD PL_[LITRL] INST. TO DCS
                        00C7  4272          LOOP            J,1,6                       ; LOOP ON J FOR 6 TEST PATTERNS
                        00CE  4273          SA01PAT         J,1$                        ; GENERATE A SA01 TEST PATTERN
                        00D6  4274          LOADREG         WDREG,1$,,L                 ; LOAD WDREG WITH TEST PATTERN
                        00DE  4275          ERRLOOP                                     ; LOOP ON ERROR
                        00E0  4276          FETCH           0                           ; EXECUTE THE ROT/RL.MM.P AND
                        00E5  4277          BRSTCLK         4                           ; WRITE THE RESULT TO WHREG
                        00E9  4278          CMPREG          WHREG,4$,K,L                ; IS THE WHREG CORRECT?
                        00F2  4279          IFERROR         ,<<SRM><SRK>>               ; IF NOT, SIGNAL AN ERROR
                        00F9  4280          INCINDEX        K                           ; INCREMENT K FOR NEXT PATTERN
                        00FC  4281          ENDLOOP         J                           ; CONTINUE FOR 6 TEST PATTERNS
                        00FF  4282          ENDLOOP         I                           ; CONTINUE FOR 7 PLATCH VALUES
                        0102  4283
                        0102  4284          SUBTEST         3
                        0105
                        0105              ;////////////////////////////////////////////////////////////////////////////
                        0105  4285 ;+
                        0105  4286 ; This subtest examines the ROT/RL.MM.PTE micro-order.
                        0105  4287 ;-
                        0105  4288          LOADDCS         12$,,3,1                    ; LOAD ROT/RL.MM.PTE INST.
                        010C  4289          LDFIELD         7$,,W,10$,31,39             ; LOAD THE LITRL FIELD WITH 0
                        011D  4290          LOADDCS         10$,,1,1                    ; LOAD PL_[0] INTO DCS
                        0124  4291          LOOP            I,1,6                       ; LOOP ON J FOR 6 TEST PATTERNS
                        012B  4292          SA01PAT         I,1$                        ; GENERATE A SA01 PATTERN
                        0133  4293          LOADREG         WDREG,1$,,L                 ; LOAD PATTERN INTO WDREG
                        013B  4294          ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                        013D  4295          FETCH           0                           ; EXECUTE ROT/RL.MM.PTE AND
                        0142  4296          BRSTCLK         4                           ; WRITE THE RESULT TO WHREG
                        0146  4297          CMPREG          WHREG,6$,I,L                ; IS THE WHREG CORRECT?
                        014F  4298          IFERROR         ,<<SRK><SRM><ALP><ALK>>     ; IF NOT, SIGNAL AN ERROR
                        0158  4299          ENDLOOP         I                           ; CONTINUE FOR 6 TEST PATTERN
                        015B  4300          SKIP                                        ; SKIP TO NEXT TEST
                        0162  4301
                        0162  4302 BEGIN_TEST_DATA
                        0162
                        0162              ;-------------------------------------------------------------------------
                        0162  4303
                00000000  0162  4304 1$:      .LONG   0       ; LOCATION TO HOLD SA01PAT FOR WDREG
                        0166  4305
                        0166  4306 ; Expected data for Subtest 1
                        0166  4307
33333333 55555555 AAAAAAAA  0166  4308 2$:      .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 0   K = 01-03
0000FFFF 00FF00FF 0F0F0F0F  0172  4309          .LONG   ^X0F0F0F0F,^X00FF00FF,^X0000FFFF ;               K = 04-06
66666666 AAAAAAAA 55555555  017E  4310          .LONG   ^X55555555,^XAAAAAAAA,^X66666666 ; PLATCH = 1   K = 07-09
0001FFFE 01FE01FE 1E1E1E1E  018A  4311          .LONG   ^X1E1E1E1E,^X01FE01FE,^X0001FFFE ;               K = 0A-0C
CCCCCCCC 55555555 AAAAAAAA  0196  4312          .LONG   ^XAAAAAAAA,^X55555555,^XCCCCCCCC ; PLATCH = 2   K = 0D-0F
0003FFFC 03FC03FC 3C3C3C3C  01A2  4313          .LONG   ^X3C3C3C3C,^X03FC03FC,^X0003FFFC ;               K = 10-12
99999999 AAAAAAAA 55555555  01AE  4314          .LONG   ^X55555555,^XAAAAAAAA,^X99999999 ; PLATCH = 3   K = 13-15
0007FFF8 07F807F8 78787878  01BA  4315          .LONG   ^X78787878,^X07F807F8,^X0007FFF8 ;               K = 16-18
                        01C6  4316
```

```
                                 01C6  4317 ; PLATCH values for Subtest 1
                                 01C6  4318
                           0000  01C6  4319 3$:     .WORD   0       ; I = 1
                           0001  01C8  4320         .WORD   1       ; I = 2
                           0002  01CA  4321         .WORD   2       ; I = 3
                           0003  01CC  4322         .WORD   3       ; I = 4
                                 01CE  4323
                                 01CE  4324 ; Expected data for Subtest 2
                                 01CE  4325
33333333 55555555 AAAAAAAA       01CE  4326 4$:     .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 4   K = 01-03
000FFFF0 0FF00FF0 F0F0F0F0       01DA  4327         .LONG   ^XF0F0F0F0,^X0FF00FF0,^X000FFFF0 ;               K = 04-06
33333333 55555555 AAAAAAAA       01E6  4328         .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 8   K = 07-09
00FFFF00 FF00FF00 0F0F0F0F       01F2  4329         .LONG   ^X0F0F0F0F,^XFF00FF00,^X00FFFF00 ;               K = 0A-0C
33333333 55555555 AAAAAAAA       01FE  4330         .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 12  K = 0D-0F
0FFFF000 F00FF00F F0F0F0F0       020A  4331         .LONG   ^XF0F0F0F0,^XF00FF00F,^X0FFFF000 ;               K = 10-12
33333333 55555555 AAAAAAAA       0216  4332         .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 16  K = 13-15
FFFF0000 00FF00FF 0F0F0F0F       0222  4333         .LONG   ^X0F0F0F0F,^X00FF00FF,^XFFFF0000 ;               K = 16-18
33333333 55555555 AAAAAAAA       022E  4334         .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 20  K = 19-1B
FFF0000F 0FF00FF0 F0F0F0F0       023A  4335         .LONG   ^XF0F0F0F0,^X0FF00FF0,^XFFF0000F ;               K = 1C-1E
33333333 55555555 AAAAAAAA       0246  4336         .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 24  K = 1F-21
FF0000FF FF00FF00 0F0F0F0F       0252  4337         .LONG   ^X0F0F0F0F,^XFF00FF00,^XFF0000FF ;               K = 22-23
33333333 55555555 AAAAAAAA       025E  4338         .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 28  K = 24-26
F0000FFF F00FF00F F0F0F0F0       026A  4339         .LONG   ^XF0F0F0F0,^XF00FF00F,^XF0000FFF ;               K = 27-29
                                 0276  4340
                                 0276  4341 ; PLATCH values for Subtest 2
                                 0276  4342
                           0004  0276  4343 5$:     .WORD   4       ; I = 1
                           0008  0278  4344         .WORD   8       ; I = 2
                           000C  027A  4345         .WORD   12      ; I = 3
                           0010  027C  4346         .WORD   16      ; I = 4
                           0014  027E  4347         .WORD   20      ; I = 5
                           0018  0280  4348         .WORD   24      ; I = 6
                           001C  0282  4349         .WORD   28      ; I = 7
                                 0284  4350
                                 0284  4351 ; Expected data for Subtest 3
                                 0284  4352
66666666 AAAAAAAA 55555555       0284  4353 6$:     .LONG   ^X55555555,^XAAAAAAAA,^X66666666 ; I = 01-03
01FFFE00 FE01FE01 1E1E1E1E       0290  4354         .LONG   ^X1E1E1E1E,^XFE01FE01,^X01FFFE00 ; I = 04-06
                                 029C  4355
                                 029C  4356 ; PLATCH value for Subtest 3
                                 029C  4357
                           0000  029C  4358 7$:     .WORD   0
                                 029E  4359
                                 029E  4360 ; Microword to load the PLATCH
                                 029E  4361
                                 029E  4362 10$:
U 01, 7700,1800,EF64,0300,0470,1802  029E  4363 ;% 01   PL_[0]  ; PLATCH GETS THE CONTENTS OF THE LITRL FIELD
                                 02A9  4367
                                 02A9  4368 ; Microword to execute a left rotation of the MBUS by the PLATCH
                                 02A9  4369
                                 02A9  4370 11$:
U 03, 6700,8800,8747,0300,0470,1804  02A9  4371 ;% 03   MSRC/TEMP0,ROT/RL.MM.P,ALPCTL/WX_S.Q_0,RDM_WB
                                 02B4  4375
                                 02B4  4376 ; Microword to execute a left rotation of the MBUS by 9 bits (A PTE)
                                 02B4  4377
```

```
                                    02B4  4378 12$:
U 03, 6700,8800,4747,0300,0470,1804 02B4  4379 ;% 03   MSRC/TEMP0,ROT/RL.MM.PTE,ALPCTL/WX_S.Q_0,RDM_WB
                                    02BF  4383
                                    02BF  4384 BEGIN_CPU_DATA
                                    0002  4385
                                    0002  4386 DCS_DATA
                                    0001  4387
U 00, 7700,C800,0364,0300,0470,1801 0001  4388 ;% 00   NOP
U 01, 7700,1800,EF64,0300,0470,1802 000C  4392 ;% 01   PL_[0]
U 02, 5700,4860,0364,0300,0470,1803 0017  4396 ;% 02   M[TEMP0]_RDM
U 03, 6700,8800,8747,0300,0470,1804 0022  4400 ;% 03   MSRC/TEMP0,ROT/RL.MM.P,ALPCTL/WX_S.Q_0,RDM_WB
U 04, 7300,4800,0364,0300,0470,1805 002D  4404 ;% 04   STOP.CLOCK
                                    0038  4408
                                    0038  4409 RTEMP_DATA
                                    0001  4410
                           00000000 0001  4411          .LONG   0       ; INITIALIZES MTEMP0
                                    0005  4412
                                    0005  4413 END_CPU_DATA
                                    02BF  4414
                                    02BF  4415 END_TEST_DATA
                                    02BF
                                    02BF        ;-------------------------------------------------------------------------
                                    02BF  4416
                                    02BF  4417 ENDTEST
```

```
                        0044        .SBTTL  "TEST   04F     Rotator Test 29 Right Rotation of MBUS concat. with RBUS Tes
                        0044  4419 ;*******************************************************************************
                        0044  4420 ;++
                        0044  4421 ;
                        0044  4422 ; FUNCTIONAL DESCRIPTION
                        0044  4423 ;
                        0044  4424 ;       This test examines the logic that performs a right rotation of the MBUS
                        0044  4425 ;       concatenated with the RBUS. The test loads specific data first into the
                        0044  4426 ;       PLATCH and then the SLATCH to rotate M'R (MBUS concatenated with RBUS)
                        0044  4427 ;       the number of bits necessary to check out the shift logic. This logic
                        0044  4428 ;       is inside the ALP and SRM gate arrays which in turn is controlled by
                        0044  4429 ;       logic inside the SRK and ALK chips.
                        0044  4430 ;
                        0044  4431 ; TEST ALGORITHM
                        0044  4432 ;
                        0044  4433 ;       This test is implemented in four subtests.
                        0044  4434 ;
                        0044  4435 ;       Subtests 1 and 3
                        0044  4436 ;
                        0044  4437 ;       These subtests examine the ROT/RR.MR.P and ROT/RR.MR.S micro-orders for
                        0044  4438 ;       PLATCH and SLATCH values of 0,1,2 and 3. This will check out the second
                        0044  4439 ;       level shifter inside the ALP gate array which receives data from the
                        0044  4440 ;       SRM chips and is controlled by signals from the ALK and SRK chips.
                        0044  4441 ;
                        0044  4442 ;       Subtests 2 and 4
                        0044  4443 ;
                        0044  4444 ;       These subtests examine the ROT/RR.MR.P and ROT/RR.MR.S micro-orders for
                        0044  4445 ;       PLATCH and SLATCH values of 4,8,12,16,20,24 and 28. This will check out
                        0044  4446 ;       the first level shifter inside the SRM gate array, controlled by SRK.
                        0044  4447 ;
                        0044  4448 ;       The test algorithm for each subtest is:
                        0044  4449 ;
                        0044  4450 ;        1. Initialize the CPU. (Subtest 1 only)
                        0044  4451 ;        2. Initialize the index K with a 1.
                        0044  4452 ;        2. Start signature analysis.
                        0044  4453 ;        3. Loop on I for the number of latch values to be tested.
                        0044  4454 ;        4. Load the LITRL field of a PL_[0] (Subtest 2) or SL_[0] (Subtest 3)
                        0044  4455 ;           micro-instruction with a value from a table indexed by I.
                        0044  4456 ;        5. Load the modified micro-instruction into DCS.
                        0044  4457 ;        6. Load DCS with the test micro-instruction to rotate right M'R by the
                        0044  4458 ;           number specified in the PLATCH for Subtests 1 and 2 (ROT/RR.MR.P)
                        0044  4459 ;           or to rotate right M'R by the number specified in the SLATCH for
                        0044  4460 ;           Subtests 3 and 4 (ROT/RR.MR.S). Both of these instructions have the
                        0044  4461 ;           MSRC and RSRC fields set up to source TEMP0.
                        0044  4462 ;        7. Loop on J for the 6 SA01 test patterns to be examined.
                        0044  4463 ;        8. Generate a stuck at zero or one test pattern.
                        0044  4464 ;        9. Load the generated test pattern into the WDREG.
                        0044  4465 ;       10. Loop on error from here.
                        0044  4466 ;       11. Execute the micro-code to:
                        0044  4467 ;               1. Load the PLATCH (Subtests 1 and 2) or SLATCH (Subtests 3 and
                        0044  4468 ;                  4) with the value generated by step 4.
                        0044  4469 ;               2. Load the scratchpad R[TEMP0] with the contents of WDREG.
                        0044  4470 ;               3. Execute the rotation ROT/RR.MR.P (Subtests 1 and 2) or
                        0044  4471 ;                  ROT/RR.MR.S (Subtests 3 and 4) writing the result to WHREG.
                        0044  4472 ;       12. Compare the contents of the WHREG against the expected data from
```

```
0044  4473 ;          a table indexed by K.
0044  4474 ;       13. If the expected data and WHREG do not agree call out an error.
0044  4475 ;       14. Increment the index K for the next expected data pattern.
0044  4476 ;       15. Loop back to step 7 for all 6 latch values.
0044  4477 ;       16. Loop on J for 1 more SA01 test pattern.
0044  4478 ;       17. Load the MSRC field with TEMP1 and the RSRC field with TEMP2 to
0044  4479 ;           source the last SA01 test pattern to the rotator.
0044  4480 ;       18. Loop on error from here.
0044  4481 ;       19. Execute the same micro-code as in step 11 with the different source
0044  4482 ;           values and write the resulting rotation into the WHREG.
0044  4483 ;       20. Compare the contents of the WHREG against the expected data from
0044  4484 ;           a table indexed by K.
0044  4485 ;       21. If the expected data and WHREG do not agree call out an error.
0044  4486 ;       22. Increment the index K for the next expected data pattern.
0044  4487 ;       23. End the loop started at step 16.
0044  4488 ;       24. Continue the loop on I for the other latch values.
0044  4489 ;       25. Stop signature analysis.
0044  4490 ;
0044  4491 ; TEST DATA
0044  4492 ;
0044  4493 ;       The data patterns input to the rotator over the RBUS are the standard
0044  4494 ;       stuck at zero or one test pattterns used in most register tests.
0044  4495 ;       The expected data consists of the same data rotated right the proper
0044  4496 ;       number of bits. By using these data patterns it is possible to find any
0044  4497 ;       stuck bit in the rotator data path as well as any bit tied to any other
0044  4498 ;       bit, even after it has undergone a rotation.
0044  4499 ;
0044  4500 ; LOGIC DESCRIPTION
0044  4501 ;
0044  4502 ;       The logic involved in this test includes the first and second level
0044  4503 ;       shifters which are internal to the SRM and ALP gate arrays. The right
0044  4504 ;       rotation function is decoded from the ROT field of the micro-word by
0044  4505 ;       the SRK and ALK gate arrays also on DPM. The decoded ROT field
0044  4506 ;       generates the control signals SHF1L, SHF0L, PRI1 and PRI0 that control
0044  4507 ;       the SRM and ALP.
0044  4508 ;
0044  4509 ; ASSUMPTIONS
0044  4510 ;
0044  4511 ;       This test assumes that the MBUS, RBUS, SBUS, WBUS, the scratchpads
0044  4512 ;       RTEMP0, MTEMP1 and RTEMP2, the Alu's WMUX_SBUS data path have all been
0044  4513 ;       tested prior to the execution of this test.
0044  4514 ;
0044  4515 ; ERROR DESCRIPTION
0044  4516 ;
0044  4517 ;       On error Subtests 1 and 3 will call out the ALP,SRM,SRK and ALK chips.
0044  4518 ;       On error Subtests 2 and 4 will call out the SRM and SRK chips.
0044  4519 ;
0044  4520 ;       All subtests will also type out the following data:
0044  4521 ;
0044  4522 ;       EXPECTED WBUS DATA
0044  4523 ;       RECEIVED WBUS DATA
0044  4524 ;       THE INDEX FOR THE PLATCH OR SLATCH DATA TABLE (INDEX I)
0044  4525 ;       THE INDEX FOR THE TEST PATTERN BEING USED (INDEX J)
0044  4526 ;       THE INDEX FOR THE EXPECTED DATA TABLE (INDEX K)
0044  4527 ;--
```

```
                     0044  4528 ;*******************************************************************************
                     0044  4529
                     0044  4530         SUBTEST         1
                     0047
                     0047       ;////////////////////////////////////////////////////////////////////////////
                     0047  4531 ;+
                     0047  4532 ; This subtest examines function ROT/RR.MR.P for PLATCH = 0,1,2,3.
                     0047  4533 ;-
                     0047  4534         INITIALIZE                              ; INITIALIZE THE CPU
                     0049  4535         LDINDEX         K,1                     ; INITIALIZE INDEX K WITH 1
                     0050  4536         LOOP            I,1,4                   ; LOOP ON I FOR 4 PLATCH VALUES
                     0057  4537         LDFIELD         2$,I,W,10$,31,39        ; LOAD LITRL FIELD WITH VALUE
                     0068  4538         LOADDCS         10$,,1,1                ; LOAD PL_[LITRL] INTO DCS
                     006F  4539         LOADDCS         12$,,3,1                ; LOAD ROT/RR.MR.P MICRO-ORDER
                     0076  4540         LOOP            J,1,6                   ; LOOP ON J FOR 6 TEST PATTERNS
                     007D  4541         SA01PAT         J,1$                    ; GENERATE A SA01 PATTERN
                     0085  4542         LOADREG         WDREG,1$,,L             ; LOAD PATTERN INTO WDREG
                     008D  4543         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                     008F  4544         FETCH           0                       ; EXECUTE THE ROT/RR.MR.P ORDER
                     0094  4545         BRSTCLK         4                       ; AND WRITE RESULT INTO WHREG
                     0098  4546         CMPREG          WHREG,3$,K,L            ; IS WHREG CORRECT?
                     00A1  4547         IFERROR         ,<<SRK><SRM>>           ; IF NOT, SIGNAL AN ERROR
                     00A8  4548         INCINDEX        K                       ; INCREMENT K FOR NEXT PATTERN
                     00AB  4549         ENDLOOP         J                       ; CONTINUE FOR 6 TEST PATTERNS
                     00AE  4550         LOOP            J,7,7                   ; LOOP ON J FOR 7TH PATTERN
                     00B5  4551         LOADDCS         13$,,3,1                ; LOAD MSRC/TEMP1,RSRC/TEMP2
                     00BC  4552         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                     00BE  4553         FETCH           0                       ; EXECUTE THE ROT/RR.MR.P ORDER
                     00C3  4554         BRSTCLK         4                       ; AND WRITE RESULT TO WHREG
                     00C7  4555         CMPREG          WHREG,3$,K,L            ; IS THE WHREG CORRECT?
                     00D0  4556         IFERROR         ,<<SRK><SRM>>           ; IF NOT, SIGNAL AN ERROR
                     00D7  4557         INCINDEX        K                       ; INCREMENT K FOR NEXT PATTERN
                     00DA  4558         ENDLOOP         J                       ; END LOOP FOR 7TH PATTERN
                     00DD  4559         ENDLOOP         I                       ; CONTINUE FOR 4 PLATCH VALUES
                     00E0  4560
                     00E0  4561         SUBTEST         2
                     00E3
                     00E3       ;////////////////////////////////////////////////////////////////////////////
                     00E3  4562 ;+
                     00E3  4563 ; This subtest examines function ROT/RR.MR.P for PLATCH = 4,8,12,16,20,24,28.
                     00E3  4564 ;-
                     00E3  4565         LDINDEX         K,1                     ; INITIALIZE K WITH 1
                     00EA  4566         LOOP            I,1,7                   ; LOOP ON I FOR 7 PLATCH VALUES
                     00F1  4567         LDFIELD         4$,I,W,10$,31,39        ; LOAD LITRL FIELD WITH A VALUE
                     0102  4568         LOADDCS         10$,,1,1                ; PL_[LITRL] INSTR. TO DCS
                     0109  4569         LOADDCS         12$,,3,1                ; LOAD MSRC/TEMP0,RSRC/TEMP0
                     0110  4570         LOOP            J,1,6                   ; LOOP ON J FOR 6 TEST PATTERNS
                     0117  4571         SA01PAT         J,1$                    ; GENERATE A SA01 TEST PATTERN
                     011F  4572         LOADREG         WDREG,1$,,L             ; LOAD TEST PATTERN INTO WDREG
                     0127  4573         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                     0129  4574         FETCH           0                       ; EXECUTE ROT/RR.MR.P INST. AND
                     012E  4575         BRSTCLK         4                       ; WRITE RESULT INTO WHREG
                     0132  4576         CMPREG          WHREG,5$,K,L            ; IS WHREG CORRECT?
                     013B  4577         IFERROR         ,<<SRK><SRM>>           ; IF NOT, SIGNAL AN ERROR
                     0142  4578         INCINDEX        K                       ; INCREMENT THE INDEX K
```

```
                0145  4579          ENDLOOP         J                       ; CONTINUE FOR 6 TEST PATTERNS
                0148  4580          LOOP            J,7,7                   ; LOOP FOR 7TH TEST PATTERN
                014F  4581          LOADDCS         13$,,3,1                ; LOAD MSRC/TEMP1,RSRC/TEMP2
                0156  4582          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                0158  4583          FETCH           0                       ; EXECUTE ROT/RR.MR.P INSTR.
                015D  4584          BRSTCLK         4                       ; WRITE RESULT INTO WHREG
                0161  4585          CMPREG          WHREG,5$,K,L            ; IS WHREG CORRECT?
                016A  4586          IFERROR         ,<<SRK><SRM>>           ; IF NOT, SIGNAL AN ERROR
                0171  4587          INCINDEX        K                       ; INCREMENT K FOR NEXT PATTERN
                0174  4588          ENDLOOP         J                       ; END LOOP FOR 7TH PATTERN
                0177  4589          ENDLOOP         I                       ; CONTINUE FOR 7 TEST VALUES
                017A  4590
                017A  4591          SUBTEST         3
                017D
                017D              ;////////////////////////////////////////////////////////////////////////////
                017D  4592        ;+
                017D  4593        ; This subtest examines the function ROT/RR.MR.S for SLATCH = 0,1,2,3.
                017D  4594        ;-
                017D  4595          LDINDEX         K,1                     ; INITIALIZE INDEX K WITH 1
                0184  4596          LOOP            I,1,4                   ; LOOP ON I FOR 4 SLATCH VALUES
                018B  4597          LDFIELD         2$,I,W,11$,31,39        ; LOAD LITRL FIELD WITH VALUE
                019C  4598          LOADDCS         11$,,1,1                ; LOAD SL_[LITRL] INSTR. TO DCS
                01A3  4599          LOADDCS         14$,,3,1                ; LOAD MSRC/TEMP0,RSRC/TEMP0
                01AA  4600          LOOP            J,1,6                   ; LOOP ON J FOR 6 TEST PATTERNS
                01B1  4601          SA01PAT         J,1$                    ; GENERATE A SA01 TEST PATTERN
                01B9  4602          LOADREG         WDREG,1$,,L             ; LOAD TEST PATTERN INTO WHREG
                01C1  4603          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                01C3  4604          FETCH           0                       ; EXECUTE A ROT/RR.MR.S INSTR.
                01C8  4605          BRSTCLK         4                       ; AND WRITE RESULT INTO WHREG
                01CC  4606          CMPREG          WHREG,3$,K,L            ; IS WHREG CORRECT?
                01D5  4607          IFERROR         ,<<SRK><SRM>>           ; IF NOT, SIGNAL AN ERROR
                01DC  4608          INCINDEX        K                       ; INCREMENT K FOR NEXT PATTERN
                01DF  4609          ENDLOOP         J                       ; CONTINUE FOR 6 TEST PATTERNS
                01E2  4610          LOOP            J,7,7                   ; LOOP ON J FOR 7TH PATTERN
                01E9  4611          LOADDCS         15$,,3,1                ; LOAD MSRC/TEMP1,RSRC/TEMP2
                01F0  4612          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                01F2  4613          FETCH           0                       ; EXECUTE ROT/RR.MR.S INSTR.
                01F7  4614          BRSTCLK         4                       ; AND WRITE RESULT INTO WHREG
                01FB  4615          CMPREG          WHREG,3$,K,L            ; IS WHREG CORRECT?
                0204  4616          IFERROR         ,<<SRK><SRM>>           ; IF NOT, SIGNAL AN ERROR
                020B  4617          INCINDEX        K                       ; INCREMENT K FOR NEXT PATTERN
                020E  4618          ENDLOOP         J                       ; END LOOP FOR 7TH PATTERN
                0211  4619          ENDLOOP         I                       ; CONTINUE FOR 4 SLATCH VALUES
                0214  4620
                0214  4621          SUBTEST         4
                0217
                0217              ;////////////////////////////////////////////////////////////////////////////
                0217  4622        ;+
                0217  4623        ; This subtest examines function ROT/RR.MR.S for SLATCH = 4,8,12,16,20,24,28.
                0217  4624        ;-
                0217  4625          LDINDEX         K,1                     ; INITIALIZE THE INDEX K
                021E  4626          LOOP            I,1,7                   ; LOOP ON I FOR 7 SLATCH VALUES
                0225  4627          LDFIELD         4$,I,W,11$,31,39        ; LOAD LITRL WITH SLATCH VALUE
                0236  4628          LOADDCS         11$,,1,1                ; LOAD SL_[LITRL] INTO DCS
                023D  4629          LOADDCS         14$,,3,1                ; LOAD RSRC/TEMP0,MSRC/TEMP0
```

```
                                0244          LOOP          J,1,6                    ; LOOP ON J FOR 6 TEST PATTERNS
                                024B          SA01PAT       J,1$                     ; GENERATE A SA01 TEST PATTERN
                                0253          LOADREG       WDREG,1$,,L              ; LOAD PATTERN INTO WDREG
                                025B          ERRLOOP                                ; LOOP ON ERROR FROM HERE
                                025D          FETCH         0                        ; EXECUTE THE ROT/RR.MR.S ORDER
                                0262          BRSTCLK       4                        ; AND WRITE RESULT TO WHREG
                                0266          CMPREG        WHREG,5$,K,L             ; IS THE WHREG CORRECT?
                                026F          IFERROR       ,<<SRK><SRM>>            ; IF NOT, SIGNAL AN ERROR
                                0276          INCINDEX      K                        ; INCREMENT K FOR NEXT PATTERN
                                0279          ENDLOOP       J                        ; CONTINUE FOR 6 TEST PATTERNS
                                027C          LOOP          J,7,7                    ; LOOP ON J FOR 7TH PATTERN
                                0283          LOADDCS       15$,,3,1                 ; LOAD RSRC/TEMP1,MSRC/TEMP2
                                028A          ERRLOOP                                ; LOOP ON ERROR FROM HERE
                                028C          FETCH         0                        ; EXECUTE ROT/RR.MR.S ON 7TH
                                0291          BRSTCLK       4                        ; PATTERN AND WRITE TO WHREG
                                0295          CMPREG        WHREG,5$,K,L             ; IS THE WHREG CORRECT?
                                029E          IFERROR       ,<<SRK><SRM>>            ; IF NOT, SIGNAL AN ERROR
                                02A5          INCINDEX      K                        ; INCREMENT K FOR NEXT PATTER
                                02A8          ENDLOOP       J                        ; END LOOP FOR 7TH PATTERN
                                02AB          ENDLOOP       I                        ; CONTINUE FOR 7 PLATCH VALUES
                                02AE          SKIP                                   ; SKIP TO NEXT TEST
                                02B5
                                02B5  BEGIN_TEST_DATA
                                02B5
                                02B5  ;-----------------------------------------------------------------------
                                02B5
                     00000000   02B5  1$:     .LONG   0       ; LOCATION TO HOLD SA01PAT FOR WDREG
                                02B9
                                02B9  ; Table of PLATCH values for Subtest 1 and SLATCH values for Subtest 3.
                                02B9
                         0000   02B9  2$:     .WORD   0       ; I = 1
                         0001   02BB          .WORD   1       ; I = 2
                         0002   02BD          .WORD   2       ; I = 3
                         0003   02BF          .WORD   3       ; I = 4
                                02C1
                                02C1  ; Expected Data for Subtests 1 and 3
                                02C1  ; The numbers to the right indicate the LATCH and K index values.
                                02C1
33333333 55555555 AAAAAAAA      02C1  3$:     .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; LATCH = 0    K = 01-03
0000FFFF 00FF00FF 0F0F0F0F      02CD          .LONG   ^X0F0F0F0F,^X00FF00FF,^X0000FFFF ;              K = 04-06
                  FFFFFFFF      02D9          .LONG   ^XFFFFFFFF                       ;              K = 07
99999999 AAAAAAAA 55555555      02DD          .LONG   ^X55555555,^XAAAAAAAA,^X99999999 ; LATCH = 1    K = 08-0A
80007FFF 807F807F 87878787      02E9          .LONG   ^X87878787,^X807F807F,^X80007FFF ;              K = 0B-0D
                  7FFFFFFF      02F5          .LONG   ^X7FFFFFFF                       ;              K = 0E
CCCCCCCC 55555555 AAAAAAAA      02F9          .LONG   ^XAAAAAAAA,^X55555555,^XCCCCCCCC ; LATCH = 2    K = 0F-11
C0003FFF C03FC03F C3C3C3C3      0305          .LONG   ^XC3C3C3C3,^XC03FC03F,^XC0003FFF ;              K = 12-14
                  3FFFFFFF      0311          .LONG   ^X3FFFFFFF                       ;              K = 15
66666666 AAAAAAAA 55555555      0315          .LONG   ^X55555555,^XAAAAAAAA,^X66666666 ; LATCH = 3    K = 16-18
E0001FFF E01FE01F E1E1E1E1      0321          .LONG   ^XE1E1E1E1,^XE01FE01F,^XE0001FFF ;              K = 19-1B
                  1FFFFFFF      032D          .LONG   ^X1FFFFFFF                       ;              K = 1C
                                0331
                                0331  ; Table of PLATCH values for Subtest 2 and SLATCH values for Subtest 4
                                0331
                         0004   0331  4$:     .WORD   4       ; I = 1
                         0008   0333          .WORD   8       ; I = 2
```

```
                        000C  0335  4683          .WORD   12      ; I = 3
                        0010  0337  4684          .WORD   16      ; I = 4
                        0014  0339  4685          .WORD   20      ; I = 5
                        0018  033B  4686          .WORD   24      ; I = 6
                        001C  033D  4687          .WORD   28      ; I = 7
                              033F  4688
                              033F  4689 ; Expected Data for Subtests 2 and 4
                              033F  4690 ; The numbers to the right indicate the LATCH and K index values.
                              033F  4691
        33333333 55555555 AAAAAAAA  033F  4692 5$:     .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; LATCH = 4    K = 01-03
        F0000FFF F00FF00F F0F0F0F0  034B  4693          .LONG   ^XF0F0F0F0,^XF00FF00F,^XF0000FFF ;               K = 04-06
                           0FFFFFFF  0357  4694          .LONG   ^X0FFFFFFF                       ;               K = 07
        33333333 55555555 AAAAAAAA  035B  4695          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; LATCH = 8    K = 08-0A
        FF0000FF FF00FF00 0F0F0F0F  0367  4696          .LONG   ^X0F0F0F0F,^XFF00FF00,^XFF0000FF ;               K = 0B-0D
                           00FFFFFF  0373  4697          .LONG   ^X00FFFFFF                       ;               K = 0E
        33333333 55555555 AAAAAAAA  0377  4698          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; LATCH = 12   K = 0F-11
        FFF0000F 0FF00FF0 F0F0F0F0  0383  4699          .LONG   ^XF0F0F0F0,^X0FF00FF0,^XFFF0000F ;               K = 12-14
                           000FFFFF  038F  4700          .LONG   ^X000FFFFF                       ;               K = 15
        33333333 55555555 AAAAAAAA  0393  4701          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; LATCH = 16   K = 16-18
        FFFF0000 00FF00FF 0F0F0F0F  039F  4702          .LONG   ^X0F0F0F0F,^X00FF00FF,^XFFFF0000 ;               K = 19-1B
                           0000FFFF  03AB  4703          .LONG   ^X0000FFFF                       ;               K = 1C
        33333333 55555555 AAAAAAAA  03AF  4704          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; LATCH = 20   K = 1D-1F
        0FFFF000 F00FF00F F0F0F0F0  03BB  4705          .LONG   ^XF0F0F0F0,^XF00FF00F,^X0FFFF000 ;               K = 20-22
                           00000FFF  03C7  4706          .LONG   ^X00000FFF                       ;               K = 23
        33333333 55555555 AAAAAAAA  03CB  4707          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; LATCH = 24   K = 24-26
        00FFFF00 FF00FF00 0F0F0F0F  03D7  4708          .LONG   ^X0F0F0F0F,^XFF00FF00,^X00FFFF00 ;               K = 27-29
                           000000FF  03E3  4709          .LONG   ^X000000FF                       ;               K = 2A
        33333333 55555555 AAAAAAAA  03E7  4710          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; LATCH = 28   K = 2B-2D
        000FFFF0 0FF00FF0 F0F0F0F0  03F3  4711          .LONG   ^XF0F0F0F0,^X0FF00FF0,^X000FFFF0 ;               K = 2E-30
                           0000000F  03FF  4712          .LONG   ^X0000000F                       ;               K = 31
                              0403  4713
                              0403  4714 ; Microwords to load the PLATCH and the SLATCH
                              0403  4715
                              0403  4716 10$:
U 01, 7700,1800,EF64,0300,0470,1802   0403  4717 ;% 01   PL_[0] ; LOAD THE PLATCH WITH THE CONTENTS OF THE LITRL FIELD
                              040E  4721
                              040E  4722 11$:
U 01, 7700,1800,F764,0300,0470,1802   040E  4723 ;% 01   SL_[0] ; LOAD THE SLATCH WITH THE CONTENTS OF THE LITRL FIELD
                              0419  4727
                              0419  4728 ; Test microword for Subtest 1
                              0419  4729 ;
                              0419  4730 ; This microword tests the right rotation of M'R by the PLATCH for six stuck at
                              0419  4731 ; zero or one test patterns.
                              0419  4732
                              0419  4733 12$:
U 03, 6700,C800,1347,0300,0470,1804   0419  4734 ;% 03   MSRC/TEMP0,RSRC/TEMP0,ROT/RR.MR.P,ALPCTL/WX_S.Q_0,RDM_WB
                              0424  4738
                              0424  4739 ; Test microword for Subtest 2
                              0424  4740 ;
                              0424  4741 ; This microword tests the right rotation of M'R by the PLATCH for the seventh
                              0424  4742 ; stuck at zero or one test pattern.
                              0424  4743
                              0424  4744 13$:
U 03, 6700,C801,1347,0308,0470,1804   0424  4745 ;% 03   MSRC/TEMP1,RSRC/TEMP2,ROT/RR.MR.P,ALPCTL/WX_S.Q_0,RDM_WB
                              042F  4749
```

```
                                042F  4750 ; Test microword for Subtest 3
                                042F  4751 ;
                                042F  4752 ; This microword tests the right rotation of M'R by the SLATCH for six stuck at
                                042F  4753 ; zero or one test patterns.
                                042F  4754
                                042F  4755 14$:
U 03, 6700,C800,1F47,0300,0470,1804   042F  4756 ;% 03   MSRC/TEMP0,RSRC/TEMP0,ROT/RR.MR.S,ALPCTL/WX_S.Q_0,RDM_WB
                                043A  4760
                                043A  4761 ; Test microword for Subtest 4
                                043A  4762 ;
                                043A  4763 ; This microword tests the right rotation of M'R by the SLATCH for the seventh
                                043A  4764 ; stuck at zero or one test pattern.
                                043A  4765
                                043A  4766 15$:
U 03, 6700,C801,1F47,0308,0470,1804   043A  4767 ;% 03   MSRC/TEMP1,RSRC/TEMP2,ROT/RR.MR.S,ALPCTL/WX_S.Q_0,RDM_WB
                                0445  4771
                                0445  4772 BEGIN_CPU_DATA
                                0002  4773
                                0002  4774 DCS_DATA
                                0001  4775
U 00, 7700,C800,0364,0300,0470,1801   0001  4776 ;% 00   NOP
U 01, 7700,1800,EF64,0300,0470,1802   000C  4780 ;% 01   PL_[0]
U 02, 5700,0840,0364,0300,0470,1803   0017  4784 ;% 02   R[TEMP0]_RDM
U 03, 6700,C800,1347,0300,0470,1804   0022  4788 ;% 03   MSRC/TEMP0,RSRC/TEMP0,ROT/RR.MR.P,ALPCTL/WX_S.Q_0,RDM_WB
U 04, 7300,4800,0364,0300,0470,1805   002D  4792 ;% 04   STOP.CLOCK
                                0038  4796
                                0038  4797 RTEMP_DATA
                                0001  4798
                    00000000    0001  4799          .LONG   0               ; INITIALIZES RTEMP0 AND MTEMP0
                    00000000    0005  4800          .LONG   0               ; CONSTANT 0 FOR MTEMP1
                    FFFFFFFF    0009  4801          .LONG   ^XFFFFFFFF      ; CONSTANT 1 FOR RTEMP2
                                000D  4802
                                000D  4803 END_CPU_DATA
                                0445  4804
                                0445  4805 END_TEST_DATA
                                0445
                                0445      ;-------------------------------------------------------------------------------
                                0445  4806
                                0445  4807 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
                        0044        .SBTTL  "TEST   050     Rotator Test 30 Right Rotation of MBUS concat. with RBUS Tes
                        0044  4809 ;*******************************************************************************
                        0044  4810 ;++
                        0044  4811 ;
                        0044  4812 ; FUNCTIONAL DESCRIPTION
                        0044  4813 ;
                        0044  4814 ;       This test will examine the right rotation of the MBUS concatenated with
                        0044  4815 ;       the RBUS (M'R) using the SRK ALU to determine the number of bits to
                        0044  4816 ;       rotate by. This is accomplished using the rotator micro-order
                        0044  4817 ;       ROT/RR.MR.PS and rotating seven stuck at zero or one test patterns to
                        0044  4818 ;       test the rotation.
                        0044  4819 ;
                        0044  4820 ; TEST ALGORITHM
                        0044  4821 ;
                        0044  4822 ;       This test has been divided into two subtests. The first subtest exam-
                        0044  4823 ;       ines the SRK ALU by floating a zero through a field of ones in the
                        0044  4824 ;       SLATCH while keeping a constant one in the PLATCH. The second subtest
                        0044  4825 ;       examines the SRK ALU by floating a zero through a field of ones in the
                        0044  4826 ;       PLATCH while keeping a constant one in the SLATCH. Both subtests rotate
                        0044  4827 ;       a stuck at zero or one test pattern right by the number of bits
                        0044  4828 ;       specified by the sum of the SLATCH and PLATCH. The algorithm for both
                        0044  4829 ;       subtests is the same and is specified below:
                        0044  4830 ;
                        0044  4831 ;        1. Initialize the CPU. (Subtest 1 only)
                        0044  4832 ;        2. Load the LITRL field of a PL_[0] (Subtest 1) or SL_[0] (Subtest 2)
                        0044  4833 ;           micro-instruction with a constant 1.
                        0044  4834 ;        3. Load PL_[1] (Subtest 1) or SL_[1] (Subtest 2) instruction into DCS.
                        0044  4835 ;        4. Initialize the index K with a 1.
                        0044  4836 ;        5. Start signature analysis.
                        0044  4837 ;        6. Loop on I for a 5 bit floating zero pattern.
                        0044  4838 ;        7. Load the LITRL field of a SL_[0] (Subtest 1) or PL_[0] (Subtest 2)
                        0044  4839 ;           micro-instruction with a floating zero pattern from a table indexed
                        0044  4840 ;           by I and load it into DCS.
                        0044  4841 ;        8. Loop on J for 6 stuck at zero or one test patterns.
                        0044  4842 ;        9. Generate a stuck at zero or one test pattern based on index J.
                        0044  4843 ;       10. Load the WDREG with the generated test pattern.
                        0044  4844 ;       11. Loop on error from here.
                        0044  4845 ;       12. Execute the micro-code to:
                        0044  4846 ;               1. Load the PLATCH (Subtest 1) or SLATCH (Subtest 2) with a
                        0044  4847 ;                  constant 1.
                        0044  4848 ;               2. Load the SLATCH (Subtest 1) or PLATCH (Subtest 2) with a
                        0044  4849 ;                  floating zero test pattern.
                        0044  4850 ;               3. Load RTEMP0 with the contents of the WDREG.
                        0044  4851 ;               4. Execute a ROT/RR.MR.PS using RTEMP0 and MTEMP0 for the
                        0044  4852 ;                  source to rotate.
                        0044  4853 ;       13. Compare the contents of the WHREG with the expected data from a
                        0044  4854 ;           table indexed by K.
                        0044  4855 ;       14. If the WHREG is not the same as the expected data, signal an error.
                        0044  4856 ;       15. Increment the index K.
                        0044  4857 ;       16. Continue looping on J for the remaining test patterns.
                        0044  4858 ;       17. Prepare a single loop on J for the seventh test pattern.
                        0044  4859 ;       18. Load DCS with a micro-instruction to source the seventh pattern.
                        0044  4860 ;       19. Loop on error from here.
                        0044  4861 ;       20. Execute the micro-instructions to:
                        0044  4862 ;               1. Load the PLATCH (Subtest 1) or SLATCH (Subtest 2) with a
```

```
0044  4863 ;                    constant 1.
0044  4864 ;                 2. Load the SLATCH (Subtest 1) or PLATCH (Subtest 2) with a
0044  4865 ;                    floating zero test pattern.
0044  4866 ;                 3. Load RTEMP0 with the contents of the WDREG.
0044  4867 ;                 4. Execute a ROT/RR.MR.PS using R[TEMP2] and M[TEMP1] for the
0044  4868 ;                    source to rotate and write the result to WHREG.
0044  4869 ;           21. Compare the contents of the WHREG with the expected data from a
0044  4870 ;               table indexed by K.
0044  4871 ;           22. If the WHREG does not match the expected data, signal an error.
0044  4872 ;           23. Increment the index K.
0044  4873 ;           24. End the seventh loop on J.
0044  4874 ;           25. Continue looping on I until all 5 latch values have been tried.
0044  4875 ;           26. Stop signature analysis.
0044  4876 ;
0044  4877 ; TEST DATA
0044  4878 ;
0044  4879 ;           Upon execution of the ROT/RR.MR.PS micro-order, the SLATCH and the
0044  4880 ;           PLATCH are added and the resulting sum is used to right rotate the 64
0044  4881 ;           bits of MBUS concatenated with the RBUS. In Subtest 1, the SLATCH is
0044  4882 ;           loaded with the floating zero pattern, while the PLATCH is loaded with
0044  4883 ;           a constant one. For Subtest 2 the PLATCH is loaded with the floating
0044  4884 ;           zero pattern and the SLATCH is loaded with a constant one. Since there
0044  4885 ;           is no visibility into signals that control the rotator, seven stuck at
0044  4886 ;           zero or one test patterns are fed to the rotator and allowed to be
0044  4887 ;           right rotated by the sum of the S and P latches. The 32 bit result is
0044  4888 ;           checked against a longword that contains the correct result for that
0044  4889 ;           shift. The combination of the floating zero and stuck at zero or one
0044  4890 ;           test patterns are sufficient to test the ROT/RR.MR.PS micro-order.
0044  4891 ;
0044  4892 ; LOGIC DESCRIPTION
0044  4893 ;
0044  4894 ;           The logic involved in this test is contained in the SRM, SRK, and ALP
0044  4895 ;           gate arrays. SRK decodes the ROT field of the micro-word to generate
0044  4896 ;           the control signals, SHF<4:0>, PRI<1:0> and SEC<5:0> that control the
0044  4897 ;           rotation logic inside the SRM and ALP slices.
0044  4898 ;
0044  4899 ; ASSUMPTIONS
0044  4900 ;
0044  4901 ;           This test assumes the following logic has been tested and is working;
0044  4902 ;           the dual-port scratchpads TEMP0, TEMP1, and TEMP2; the MBUS, RBUS, SBUS
0044  4903 ;           WBUS, SLATCH, and PLATCH.
0044  4904 ;
0044  4905 ; ERROR DESCRIPTION
0044  4906 ;
0044  4907 ;           On error this test will call out the SRM, SRK, ALP, and ALK gate array
0044  4908 ;           and the following data:
0044  4909 ;
0044  4910 ;           EXPECTED WBUS DATA
0044  4911 ;           RECEIVED WBUS DATA
0044  4912 ;           INDEX I (SELECTS THE FLOATING ZERO PATTERN)
0044  4913 ;           INDEX J (SELECTS THE STUCK AT ZERO OR ONE TEST PATTERN)
0044  4914 ;           INDEX K (INDEX TO THE EXPECTED DATA TABLE)
0044  4915 ;
0044  4916 ;--
0044  4917 ;*******************************************************************************
```

```
                     0044  4918
                     0044  4919          SUBTEST         1
                     0047
                     0047        ;////////////////////////////////////////////////////////////////////////////////
                     0047  4920 ;+
                     0047  4921 ; This subtest floats a zero through the SLATCH keeping a constant one in the
                     0047  4922 ; PLATCH, each time rotating stuck at zero or one test patterns the number of
                     0047  4923 ; bits corresponding to the sum of the S and P latches.
                     0047  4924 ;-
                     0047  4925          INITIALIZE                                  ; INITIALIZE THE CPU
                     0049  4926          LDFIELD         4$,,W,10$,31,39             ; LOAD THE LITRL FIELD WITH A 1
                     005A  4927          LOADDCS         10$,,1,1                    ; LOAD PL_[1] INTO DCS
                     0061  4928          LDINDEX         K,1                         ; INITIALIZE INDEX K WITH A 1
                     0068  4929          LOOP            I,1,5                       ; 5 FLOATING ZERO PATTERNS
                     006F  4930          LDFIELD         3$,I,W,11$,31,39            ; LITRL GETS A FLOATING ZERO
                     0080  4931          LOADDCS         11$,,2,1                    ; LOAD SL_[LITRL] INTO DCS
                     0087  4932          LOADDCS         12$,,4,1                    ; LOAD SOURCE OF TEMP0 INTO DCS
                     008E  4933          LOOP            J,1,6                       ; 6 SA01 TEST PATTERNS
                     0095  4934          SA01PAT         J,1$                        ; GENERATE A SA01 TEST PATTERN
                     009D  4935          LOADREG         WDREG,1$,,L                 ; LOAD WDREG WITH TEST PATTERN
                     00A5  4936          ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                     00A7  4937          FETCH           0                           ; EXECUTE THE ROT/RR.MR.PS AND
                     00AC  4938          BRSTCLK         5                           ; WRITE RESULT TO WHREG
                     00B0  4939          CMPREG          WHREG,2$,K,L                ; IS WHREG CORRECT?
                     00B9  4940          IFERROR         ,<<SRK><SRM><ALP><ALK>>     ; IF NOT, SIGNAL AN ERROR
                     00C2  4941          INCINDEX        K                           ; INCREMENT K FOR NEXT PATTERN
                     00C5  4942          ENDLOOP         J                           ; CONTINUE FOR 6 SA01 PATTERNS
                     00C8  4943          LOOP            J,7,7                       ; 7TH SA01 TEST PATTERN
                     00CF  4944          LOADDCS         13$,,4,1                    ; SOURCE MTEMP1 AND RTEMP2
                     00D6  4945          ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                     00D8  4946          FETCH           0                           ; EXECUTE THE ROT/RR.MR.PS AND
                     00DD  4947          BRSTCLK         5                           ; WRITE RESULT TO WHREG
                     00E1  4948          CMPREG          WHREG,2$,K,L                ; IS WHREG CORRECT?
                     00EA  4949          IFERROR         ,<<SRK><SRM><ALP><ALK>>     ; IF NOT, SIGNAL AN ERROR
                     00F3  4950          INCINDEX        K                           ; INCREMENT K FOR NEXT PATTERN
                     00F6  4951          ENDLOOP         J                           ; END OF 7TH SA01 TEST PATTERN
                     00F9  4952          ENDLOOP         I                           ; CONTINUE 5 FLOATING PATTERNS
                     00FC  4953
                     00FC  4954          SUBTEST         2
                     00FF
                     00FF        ;////////////////////////////////////////////////////////////////////////////////
                     00FF  4955 ;+
                     00FF  4956 ; This subtest floats a zero through the PLATCH keeping a constant one in the
                     00FF  4957 ; SLATCH, each time rotating stuck at zero or one test patterns the number of
                     00FF  4958 ; bits corresponding to the sum of the S and P latches.
                     00FF  4959 ;-
                     00FF  4960          LDFIELD         4$,,W,11$,31,39             ; LOAD THE LITRL FIELD WITH A 1
                     0110  4961          LOADDCS         11$,,2,1                    ; LOAD SL_[1] INTO DCS
                     0117  4962          LDINDEX         K,1                         ; INITIALIZE THE INDEX K WITH 1
                     011E  4963          LOOP            I,1,5                       ; 5 FLOATING 0 PATTERNS
                     0125  4964          LDFIELD         3$,I,W,10$,31,39            ; LITRL FIELD GETS A FLOATING 0
                     0136  4965          LOADDCS         10$,,1,1                    ; LOAD PL_[LITRL] INTO DCS
                     013D  4966          LOADDCS         12$,,4,1                    ; LOAD SOURCE OF TEMP0 INTO DCS
                     0144  4967          LOOP            J,1,6                       ; 6 SA01 TEST PATTERNS
                     014B  4968          SA01PAT         J,1$                        ; GENERATE A SA01 TEST PATTERN
```

```
                        0153  4969          LOADREG         WDREG,1$,,L                     ; LOAD TEST PATTERN INTO WDREG
                        015B  4970          ERRLOOP                                         ; LOOP ON ERROR FROM HERE
                        015D  4971          FETCH           0                               ; EXECUTE THE ROT/RR.MR.PS AND
                        0162  4972          BRSTCLK         5                               ; WRITE THE RESULT INTO WHREG
                        0166  4973          CMPREG          WHREG,2$,K,L                    ; IS WHREG CORRECT?
                        016F  4974          IFERROR         ,<<SRK><SRM><ALP><ALK>>         ; IF NOT, SIGNAL AN ERROR
                        0178  4975          INCINDEX        K                               ; INCREMENT K FOR NEXT PATTERN
                        017B  4976          ENDLOOP         J                               ; CONTINUE FOR 6 SA01 PATTERNS
                        017E  4977          LOOP            J,7,7                           ; 7TH SA01 TEST PATTERN
                        0185  4978          LOADDCS         13$,,4,1                        ; SOURCE MTEMP1 AND RTEMP2
                        018C  4979          ERRLOOP                                         ; LOOP ON ERROR FROM HERE
                        018E  4980          FETCH           0                               ; EXECUTE THE ROT/RR.MR.PS AND
                        0193  4981          BRSTCLK         5                               ; WRITE THE RESULT INTO WHREG
                        0197  4982          CMPREG          WHREG,2$,K,L                    ; IS WHREG CORRECT?
                        01A0  4983          IFERROR         ,<<SRK><SRM><ALP><ALK>>         ; IF NOT, SIGNAL AN ERROR
                        01A9  4984          INCINDEX        K                               ; INCREMENT K FOR NEXT PATTERN
                        01AC  4985          ENDLOOP         J                               ; END 7TH SA01 TEST PATTERN
                        01AF  4986          ENDLOOP         I                               ; CONTINUE 5 FLOATING PATTERNS
                        01B2  4987          SKIP                                            ; END OF TEST
                        01B9  4988
                        01B9  4989 BEGIN_TEST_DATA
                        01B9
                        01B9                ;-------------------------------------------------------------------------
                        01B9  4990
                00000000 01B9  4991 1$:     .LONG   0       ; LOCATION TO HOLD SA01PAT FOR WDREG
                        01BD  4992 ;
                        01BD  4993 ; Expected Data Table
                        01BD  4994 ;
                        01BD  4995 ;       The comments on the right give the latch and K index values for each
                        01BD  4996 ;       longword in the table.
                        01BD  4997 ;
66666666 AAAAAAAA 55555555 01BD  4998 2$:       .LONG   ^X55555555,^XAAAAAAAA,^X66666666 ; LATCH = 31   K = 01-03
0001FFFE 01FE01FE 1E1E1E1E 01C9  4999           .LONG   ^X1E1E1E1E,^X01FE01FE,^X0001FFFE ;               K = 04-06
                  00000001 01D5  5000           .LONG   ^X00000001                       ;               K = 07
CCCCCCCC 55555555 AAAAAAAA 01D9  5001           .LONG   ^XAAAAAAAA,^X55555555,^XCCCCCCCC ; LATCH = 30   K = 08-0A
0003FFFC 03FC03FC 3C3C3C3C 01E5  5002           .LONG   ^X3C3C3C3C,^X03FC03FC,^X0003FFFC ;               K = 0B-0D
                  00000003 01F1  5003           .LONG   ^X00000003                       ;               K = 0E
33333333 55555555 AAAAAAAA 01F5  5004           .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; LATCH = 28   K = 0F-11
000FFFF0 0FF00FF0 F0F0F0F0 0201  5005           .LONG   ^XF0F0F0F0,^X0FF00FF0,^X000FFFF0 ;               K = 12-14
                  0000000F 020D  5006           .LONG   ^X0000000F                       ;               K = 15
33333333 55555555 AAAAAAAA 0211  5007           .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; LATCH = 24   K = 16-18
00FFFF00 FF00FF00 0F0F0F0F 021D  5008           .LONG   ^X0F0F0F0F,^XFF00FF00,^X00FFFF00 ;               K = 19-21
                  000000FF 0229  5009           .LONG   ^X000000FF                       ;               K = 22
33333333 55555555 AAAAAAAA 022D  5010           .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; LATCH = 16   K = 23-25
FFFF0000 00FF00FF 0F0F0F0F 0239  5011           .LONG   ^X0F0F0F0F,^X00FF00FF,^XFFFF0000 ;               K = 26-28
                  0000FFFF 0245  5012           .LONG   ^X0000FFFF                       ;               K = 29
                        0249  5013
                        0249  5014 ; SLATCH AND PLATCH values
                        0249  5015
  000F 0017 001B 001D 001E 0249  5016 3$:       .WORD   ^X1E,^X1D,^X1B,^X17,^X0F        ; FLOATING 0 TEST PATTERNS
                     0001 0253  5017 4$:        .WORD   ^X01                            ; CONSTANT 1
                        0255  5018
                        0255  5019 ; Microword to load the PLATCH
                        0255  5020
                        0255  5021 10$:
```

```
U 01, 7700,1800,EF64,0300,0470,1802  0255  5022 ;% 01   PL_[0]  ; PLATCH GETS CONTENTS OF LITRL FIELD
                                     0260  5026
                                     0260  5027 ; Microword to load the SLATCH
                                     0260  5028
                                     0260  5029 11$:
U 02, 7700,9800,F764,0300,0470,1803  0260  5030 ;% 02   SL_[0]  ; SLATCH GETS CONTENTS OF LITRL FIELD
                                     026B  5034
                                     026B  5035 ; Microword to test six SA01 patterns using the RR.MR.PS micro-order
                                     026B  5036
                                     026B  5037 12$:
U 04, 6700,0800,9347,0300,0470,1805  026B  5038 ;% 04   MSRC/TEMP0,RSRC/TEMP0,ROT/RR.MR.PS,ALPCTL/WX_S.Q_0,RDM_WB
                                     0276  5042
                                     0276  5043 ; Microword to test the seventh SA01 pattern using the RR.MR.PS micro-order
                                     0276  5044
                                     0276  5045 13$:
U 04, 6700,0801,9347,0308,0470,1805  0276  5046 ;% 04   MSRC/TEMP1,RSRC/TEMP2,ROT/RR.MR.PS,ALPCTL/WX_S.Q_0,RDM_WB
                                     0281  5050
                                     0281  5051 BEGIN_CPU_DATA
                                     0002  5052
                                     0002  5053 DCS_DATA
                                     0001  5054
U 00, 7700,C800,0364,0300,0470,1801  0001  5055 ;% 00   NOP
U 01, 7700,1800,EF64,0300,0470,1802  000C  5059 ;% 01   PL_[0]
U 02, 7700,9800,F764,0300,0470,1803  0017  5063 ;% 02   SL_[0]
U 03, 5700,8840,0364,0300,0470,1804  0022  5067 ;% 03   R[TEMP0]_RDM
U 04, 6700,0800,9347,0300,0470,1805  002D  5071 ;% 04   MSRC/TEMP0,RSRC/TEMP0,ROT/RR.MR.PS,ALPCTL/WX_S.Q_0,RDM_WB
U 05, 7300,4800,0364,0300,0470,1806  0038  5075 ;% 05   STOP.CLOCK
                                     0043  5079
                                     0043  5080 RTEMP_DATA
                                     0001  5081
                         00000000    0001  5082          .LONG   0               ; INITIALIZES RTEMP0 AND MTEMP0
                         00000000    0005  5083          .LONG   0               ; ZEROES MTEMP1 FOR 7TH TEST PATTERN
                         FFFFFFFF    0009  5084          .LONG   ^XFFFFFFFF      ; ONES IN RTEMP2 FOR 7TH TEST PATTERN
                                     000D  5085
                                     000D  5086 END_CPU_DATA
                                     0281  5087
                                     0281  5088 END_TEST_DATA
                                     0281
                                     0281       ;-----------------------------------------------------------------------
                                     0281  5089
                                     0281  5090 ENDTEST
```

```
                        0044        .SBTTL  "TEST   051     Rotator Test 31 Right Rotation of MBUS concat. with RBUS Tes
                        0044  5092 ;*******************************************************************************
                        0044  5093 ;++
                        0044  5094 ;
                        0044  5095 ; FUNCTIONAL DESCRIPTION
                        0044  5096 ;
                        0044  5097 ;       This test examines the rotator's ability to right rotate the MBUS con-
                        0044  5098 ;       catenated with the RBUS by four and nine bits by using the micro-orders
                        0044  5099 ;       ROT/RR.MR.4 and ROT/RR.MR.9.
                        0044  5100 ;
                        0044  5101 ; TEST ALGORITHM
                        0044  5102 ;
                        0044  5103 ;       This test is implemented in two subtests. The first subtest examines
                        0044  5104 ;       the ROT/RR.MR.4 micro-order, while the second subtest examines the
                        0044  5105 ;       ROT/RR.MR.9 micro-order. The algorithm used to implement both subtests
                        0044  5106 ;       is essentially the same and executes as follows:
                        0044  5107 ;
                        0044  5108 ;        1. Initialize the CPU. (Subtest 1 only)
                        0044  5109 ;        2. Load DCS with a micro-instruction that executes a ROT/RR.MR.4 (Sub-
                        0044  5110 ;           test 1) or a ROT/RR.MR.9 (Subtest 2) and sources the scratchpad
                        0044  5111 ;           TEMP0 onto both the MBUS and the RBUS.
                        0044  5112 ;        3. Start signature analysis.
                        0044  5113 ;        4. Loop on I for 6 stuck at zero or one test patterns.
                        0044  5114 ;        5. Generate a stuck at zero or one test pattern.
                        0044  5115 ;        6. Load the test pattern into the WDREG.
                        0044  5116 ;        7. Loop on error from here.
                        0044  5117 ;        8. Execute the micro-code to:
                        0044  5118 ;               1. Load RTEMP0 with the contents of the WDREG.
                        0044  5119 ;               2. Execute the rotation and write the result into WHREG.
                        0044  5120 ;        9. Compare the WHREG against the expected data, indexed by I.
                        0044  5121 ;       10. If the WHREG and the expected data do not match, signal an error.
                        0044  5122 ;       11. Continue the loop on I for the remaining test patterns.
                        0044  5123 ;       12. Load DCS with a micro-instruction that executes a ROT/RR.MR.4 (Sub-
                        0044  5124 ;           test 1) or a ROT/RR.MR.9 (Subtest 2) and sources the scratchpads
                        0044  5125 ;           MTEMP1 and RTEMP2 onto the MBUS and the RBUS.
                        0044  5126 ;       13. Setup a loop of one pass for the seventh stuck at zero or one test
                        0044  5127 ;           pattern with I = 7.
                        0044  5128 ;       14. Loop on error from here.
                        0044  5129 ;       15. Execute the micro-code to:
                        0044  5130 ;               1. Execute the rotation using MTEMP1 and RTEMP2 for the source
                        0044  5131 ;                  to rotate and write the result to WHREG.
                        0044  5132 ;       16. Compare the WHREG against the expected data, indexed by I.
                        0044  5133 ;       17. If the WHREG and the expected data do not match, signal an error.
                        0044  5134 ;       18. End the loop for the seventh test pattern.
                        0044  5135 ;       19. Stop signature analysis.
                        0044  5136 ;
                        0044  5137 ; TEST DATA
                        0044  5138 ;
                        0044  5139 ;       This test uses the stuck at zero or one test patterns required to test
                        0044  5140 ;       a quadword (64 bits). The expected data consists of these patterns
                        0044  5141 ;       rotated right by 4 bits for Subtest 1 and rotated right by 9 bits for
                        0044  5142 ;       Subtest 2. The two tables below list the values of MBUS, RBUS and WBUS
                        0044  5143 ;       for each iteration of the loop on I:
                        0044  5144 ;
                        0044  5145 ;       Subtest 1
```

```
0044  5146 ;
0044  5147 ;                    I        MBUS          RBUS          WBUS
0044  5148 ;
0044  5149 ;                    1        AAAAAAAA      AAAAAAAA      AAAAAAAA
0044  5150 ;                    2        55555555      55555555      55555555
0044  5151 ;                    3        33333333      33333333      33333333
0044  5152 ;                    4        0F0F0F0F      0F0F0F0F      F0F0F0F0
0044  5153 ;                    5        00FF00FF      00FF00FF      F00FF00F
0044  5154 ;                    6        0000FFFF      0000FFFF      F0000FFF
0044  5155 ;                    7        00000000      FFFFFFFF      0FFFFFFF
0044  5156 ;
0044  5157 ;          Subtest 2
0044  5158 ;
0044  5159 ;                    I        MBUS          RBUS          WBUS
0044  5160 ;
0044  5161 ;                    1        AAAAAAAA      AAAAAAAA      55555555
0044  5162 ;                    2        55555555      55555555      AAAAAAAA
0044  5163 ;                    3        33333333      33333333      99999999
0044  5164 ;                    4        0F0F0F0F      0F0F0F0F      87878787
0044  5165 ;                    5        00FF00FF      00FF00FF      7F807F80
0044  5166 ;                    6        0000FFFF      0000FFFF      FF80007F
0044  5167 ;                    7        00000000      FFFFFFFF      0007FFFF
0044  5168 ;
0044  5169 ; LOGIC DESCRIPTION
0044  5170 ;
0044  5171 ;       The logic involved in the right rotation by 4 bits is internal to the
0044  5172 ;       SRM bit slices and the SRK gate arrays. The logic involved in the right
0044  5173 ;       rotation by 9 bits includes the ALP and SRM slices as well as the SRK
0044  5174 ;       gate array.
0044  5175 ;
0044  5176 ; ASSUMPTIONS
0044  5177 ;
0044  5178 ;       This test assumes the following logic has been tested and is working;
0044  5179 ;       the dual-port scratchpad registers TEMP0, TEMP1, and TEMP2; the MBUS,
0044  5180 ;       RBUS and WBUS; and the ALPCTL data path for WX_S.
0044  5181 ;
0044  5182 ; ERROR DESCRIPTION
0044  5183 ;
0044  5184 ;       Subtest 1 will call out the SRM and SRK gate arrays.
0044  5185 ;       Subtest 2 will call out the SRM, ALP, and SRK gate arrays.
0044  5186 ;
0044  5187 ;       The following data will also be typed out:
0044  5188 ;
0044  5189 ;       EXPECTED WBUS DATA
0044  5190 ;       RECEIVED WBUS DATA
0044  5191 ;       LOOP COUNTER I
0044  5192 ;
0044  5193 ;--
0044  5194 ;*******************************************************************************
0044  5195
0044  5196          SUBTEST         1
0047
0047        ;////////////////////////////////////////////////////////////////////////////
0047  5197 ;+
0047  5198 ; This subtest examines the ROT/RR.MR.4 micro-order.
```

```
                     0047  5199 ;-
                     0047  5200         INITIALIZE                              ; INITIALIZE THE CPU
                     0049  5201         LOADDCS         10$,,2,1                ; SOURCE MTEMP0 AND RTEMP0
                     0050  5202         LOOP            I,1,6                   ; LOOP FOR 6 SA01 TEST PATTERNS
                     0057  5203         SA01PAT         I,1$                    ; GENERATE A SA01 TEST PATTERN
                     005F  5204         LOADREG         WDREG,1$,,L             ; LOAD TEST PATTERN INTO WDREG
                     0067  5205         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                     0069  5206         FETCH           0                       ; EXECUTE THE ROT/RR.MR.4 AND
                     006E  5207         BRSTCLK         3                       ; WRITE THE RESULT TO WHREG
                     0072  5208         CMPREG          WHREG,2$,I,L            ; IS WHREG CORRECT?
                     007B  5209         IFERROR         ,<<SRK><SRM>>           ; IF NOT, SIGNAL AN ERROR
                     0082  5210         ENDLOOP         I                       ; CONTINUE FOR 6 SA01 PATTERNS
                     0085  5211         LOADDCS         11$,,2,1                ; SOURCE MTEMP1 AND RTEMP2
                     008C  5212         LOOP            I,7,7                   ; LOOP FOR 7TH SA01 PATTERN
                     0093  5213         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                     0095  5214         FETCH           0                       ; EXECUTE THE ROT/RR.MR.4 AND
                     009A  5215         BRSTCLK         3                       ; WRITE THE RESULT TO WHREG
                     009E  5216         CMPREG          WHREG,2$,I,L            ; IS THE WHREG CORRECT?
                     00A7  5217         IFERROR         ,<<SRK><SRM>>           ; IF NOT, SIGNAL AN ERROR
                     00AE  5218         ENDLOOP         I                       ; END LOOP FOR 7TH SA01 PATTERN
                     00B1  5219
                     00B1  5220         SUBTEST         2
                     00B4
                     00B4               ;////////////////////////////////////////////////////////////////////////////
                     00B4  5221 ;+
                     00B4  5222 ; This subtest examines the ROT/RR.MR.9 micro-order.
                     00B4  5223 ;-
                     00B4  5224         LOADDCS         12$,,2,1                ; SOURCE MTEMP0 AND RTEMP0
                     00BB  5225         LOOP            I,1,6                   ; LOOP FOR 6 SA01 TEST PATTERNS
                     00C2  5226         SA01PAT         I,1$                    ; GENERATE A SA01 TEST PATTERN
                     00CA  5227         LOADREG         WDREG,1$,,L             ; LOAD THE PATTERN INTO WDREG
                     00D2  5228         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                     00D4  5229         FETCH           0                       ; EXECUTE THE ROT/RR.MR.9 AND
                     00D9  5230         BRSTCLK         3                       ; WRITE THE RESULT TO WHREG
                     00DD  5231         CMPREG          WHREG,3$,I,L            ; IS THE WHREG CORRECT?
                     00E6  5232         IFERROR         ,<<SRK><SRM><ALP>>      ; IF NOT, SIGNAL AN ERROR
                     00EE  5233         ENDLOOP         I                       ; CONTINUE FOR 6 SA01 PATTERNS
                     00F1  5234         LOADDCS         13$,,2,1                ; SOURCE MTEMP1 AND RTEMP2
                     00F8  5235         LOOP            I,7,7                   ; LOOP FOR 7TH SA01 PATTERN
                     00FF  5236         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                     0101  5237         FETCH           0                       ; EXECUTE THE ROT/RR.MR.9 AND
                     0106  5238         BRSTCLK         3                       ; WRITE THE RESULT TO WHREG
                     010A  5239         CMPREG          WHREG,3$,I,L            ; IS THE WHREG CORRECT?
                     0113  5240         IFERROR         ,<<SRK><SRM><ALP>>      ; IF NOT, SIGNAL AN ERROR
                     011B  5241         ENDLOOP         I                       ; END LOOP FOR 7TH SA01 PATTERN
                     011E  5242         SKIP                                    ; END OF TEST
                     0125  5243
                     0125  5244 BEGIN_TEST_DATA
                     0125
                     0125               ;------------------------------------------------------------------------
                     0125  5245
00000000             0125  5246 1$:      .LONG   0       ; LOCATION TO HOLD SA01PAT FOR WDREG
                     0129  5247
                     0129  5248 ; Expected data for Subtest 1
                     0129  5249
```

```
                33333333 55555555 AAAAAAAA  0129  5250 2$:     .LONG   ^XAAAAAAAA,^X55555555,^X33333333        ; I = 01-03
                F0000FFF F00FF00F F0F0F0F0  0135  5251         .LONG   ^XF0F0F0F0,^XF00FF00F,^XF0000FFF        ; I = 04-06
                                 0FFFFFFF  0141  5252         .LONG   ^X0FFFFFFF                              ; I = 07
                                           0145  5253
                                           0145  5254 ; Expected data for Subtest 2
                                           0145  5255
                99999999 AAAAAAAA 55555555  0145  5256 3$:     .LONG   ^X55555555,^XAAAAAAAA,^X99999999        ; I = 08-0A
                FF80007F 7F807F80 87878787  0151  5257         .LONG   ^X87878787,^X7F807F80,^XFF80007F        ; I = 0B-0D
                                 007FFFFF  015D  5258         .LONG   ^X007FFFFF                              ; I = 0E
                                           0161  5259
                                           0161  5260 ; Microword to test six SA01 patterns using the RR.MR.4 micro-order
                                           0161  5261
                                           0161  5262 10$:
U 02, 6700,0800,2747,0300,0470,1803        0161  5263 ;% 02   MSRC/TEMP0,RSRC/TEMP0,ROT/RR.MR.4,ALPCTL/WX_S.Q_0,RDM_WB
                                           016C  5267
                                           016C  5268 ; Microword to test seventh SA01 pattern using the RR.MR.4 micro-order
                                           016C  5269
                                           016C  5270 11$:
U 02, 6700,0801,2747,0308,0470,1803        016C  5271 ;% 02   MSRC/TEMP1,RSRC/TEMP2,ROT/RR.MR.4,ALPCTL/WX_S.Q_0,RDM_WB
                                           0177  5275
                                           0177  5276 ; Microword to test six SA01 patterns using the RR.MR.9 micro-order
                                           0177  5277
                                           0177  5278 12$:
U 02, 6700,4800,2F47,0300,0470,1803        0177  5279 ;% 02   MSRC/TEMP0,RSRC/TEMP0,ROT/RR.MR.9,ALPCTL/WX_S.Q_0,RDM_WB
                                           0182  5283
                                           0182  5284 ; Microword to test seventh SA01 pattern using the RR.MR.9 micro-order
                                           0182  5285
                                           0182  5286 13$:
U 02, 6700,4801,2F47,0308,0470,1803        0182  5287 ;% 02   MSRC/TEMP1,RSRC/TEMP2,ROT/RR.MR.9,ALPCTL/WX_S.Q_0,RDM_WB
                                           018D  5291
                                           018D  5292 BEGIN_CPU_DATA
                                           0002  5293
                                           0002  5294 DCS_DATA
                                           0001  5295
U 00, 7700,C800,0364,0300,0470,1801        0001  5296 ;% 00   NOP
U 01, 5700,8840,0364,0300,0470,1802        000C  5300 ;% 01   R[TEMP0]_RDM
U 02, 6700,0800,2747,0300,0470,1803        0017  5304 ;% 02   MSRC/TEMP0,RSRC/TEMP0,ROT/RR.MR.4,ALPCTL/WX_S.Q_0,RDM_WB
U 03, 7300,C800,0364,0300,0470,1804        0022  5308 ;% 03   STOP.CLOCK
                                           002D  5312
                                           002D  5313 RTEMP_DATA
                                           0001  5314
                                 00000000  0001  5315         .LONG   0               ; INITIALIZES RTEMP0 AND MTEMP0
                                 00000000  0005  5316         .LONG   0               ; CONSTANT ZEROES FOR MTEMP1
                                 FFFFFFFF  0009  5317         .LONG   ^XFFFFFFFF      ; CONSTANT ONES FOR RTEMP2
                                           000D  5318
                                           000D  5319 END_CPU_DATA
                                           018D  5320
                                           018D  5321 END_TEST_DATA
                                           018D
                                           018D       ;-----------------------------------------------------------------------
                                           018D  5322
                                           018D  5323 ENDTEST
```

```
                        0043        .SBTTL  "TEST   052     Rotator Test 32 Left Rotation of RBUS concat. with MBUS Test
                        0043  5325 ;*******************************************************************************
                        0043  5326 ;++
                        0043  5327 ;
                        0043  5328 ; FUNCTIONAL DESCRIPTION
                        0043  5329 ;
                        0043  5330 ;       This test examines the rotator's ability to left rotate data from the
                        0043  5331 ;       MBUS concatenated with the RBUS.
                        0043  5332 ;
                        0043  5333 ; TEST ALGORITHM
                        0043  5334 ;
                        0043  5335 ;       This test is implemented in two subtests. The first subtest examines
                        0043  5336 ;       the second level shifter inside the ALP gate arrays by executing rotat-
                        0043  5337 ;       ions of 0,1,2 and 3 bits. The second subtest examines the first level
                        0043  5338 ;       shifter inside the SRM gate array by executing rotations of 4,8,12,16,
                        0043  5339 ;       20,24 and 28 bits. The test algorithm for both is as follows:
                        0043  5340 ;
                        0043  5341 ;        1. Initialize the CPU. (Subtest 1 only)
                        0043  5342 ;        2. Initialize the index K with a 1.
                        0043  5343 ;        3. Start signature analysis.
                        0043  5344 ;        4. Loop on I for 4 (Subtest 1) or for 7 (Subtest 2) PLATCH values.
                        0043  5345 ;        5. Load the LITRL field of a PLATCH_[0] instruction with a PLATCH
                        0043  5346 ;           value from a table indexed by I.
                        0043  5347 ;        6. Load the PLATCH_[LITRL] instruction into DCS.
                        0043  5348 ;        7. Load the ROT/RL.RM.P test instruction with the source fields set
                        0043  5349 ;           to source MTEMP0 and RTEMP0.
                        0043  5350 ;        8. Loop on J for 6 stuck at zero or one test patterns.
                        0043  5351 ;        9. Generate a stuck at zero or one test pattern based on J.
                        0043  5352 ;       10. Load the WDREG with the test pattern.
                        0043  5353 ;       11. Loop on error from here.
                        0043  5354 ;       12. Execute the micro-code to:
                        0043  5355 ;               1. Load the PLATCH with the contents of the LITRL field.
                        0043  5356 ;               2. Load RTEMP0 with the contents of the WDREG.
                        0043  5357 ;               3. Execute a left rotation of the RBUS concatenated with the
                        0043  5358 ;                  MBUS using as a source RTEMP0 and MTEMP0. Write the result
                        0043  5359 ;                  to the WHREG.
                        0043  5360 ;       13. Does the contents of the WHREG match the expected data read from a
                        0043  5361 ;           table indexed by K?
                        0043  5362 ;       14. If not, signal an error.
                        0043  5363 ;       15. Increment the index K.
                        0043  5364 ;       16. Continue the loop on J.
                        0043  5365 ;       17. Loop once on J having a value of 7.
                        0043  5366 ;       18. Modify the test instruction in DCS to source MTEMP1 and RTEMP2.
                        0043  5367 ;       19. Loop on error from here.
                        0043  5368 ;       20. Execute the micro-code to:
                        0043  5369 ;               1. Load the PLATCH with the contents of the LITRL field.
                        0043  5370 ;               2. Load RTEMP0 with the contents of the WDREG.
                        0043  5371 ;               3. Execute a left rotation of the RBUS concatenated with the
                        0043  5372 ;                  MBUS using as a source RTEMP2 and MTEMP1. Write the result
                        0043  5373 ;                  to the WHREG.
                        0043  5374 ;       21. Does the contents of the WHREG match the expected data read from a
                        0043  5375 ;           table indexed by K?
                        0043  5376 ;       22. If not, signal an error.
                        0043  5377 ;       23. Increment the index K.
                        0043  5378 ;       24. End the loop on J equal to 7.
```

```
                     0043  5379 ;       25. Continue the loop on I.
                     0043  5380 ;       26. Stop signature analysis.
                     0043  5381 ;       27. Skip to next test. (Subtest 2)
                     0043  5382 ;
                     0043  5383 ; TEST DATA
                     0043  5384 ;
                     0043  5385 ;       77 data patterns are used to test the rotation. Each pattern is a stuck
                     0043  5386 ;       at zero or one test pattern rotated left the number of bits indicated
                     0043  5387 ;       by the contents of the PLATCH. There are 11 PLATCH values, 4 for
                     0043  5388 ;       Subtest 1 and 7 for Subtest 2. The PLATCH values were chosen to test
                     0043  5389 ;       the first and second level shifters inside the SRM and ALP slices.
                     0043  5390 ;
                     0043  5391 ; LOGIC DESCRIPTION
                     0043  5392 ;
                     0043  5393 ;       The logic involved in this test is contained inside the SRM and ALP
                     0043  5394 ;       slices. These slices in turn receive their control signals from the
                     0043  5395 ;       SRK gate array.
                     0043  5396 ;
                     0043  5397 ; ASSUMPTIONS
                     0043  5398 ;
                     0043  5399 ;       This test assumes the following logic has been tested and is working;
                     0043  5400 ;       the MBUS, RBUS, SBUS and WBUS; the dual port scratchpads TEMP0, TEMP1
                     0043  5401 ;       and TEMP2; the PLATCH; and the WMUX_S.Q_0 ALPCTL data path.
                     0043  5402 ;
                     0043  5403 ; ERROR DESCRIPTION
                     0043  5404 ;
                     0043  5405 ;       For Subtest 1 these gate arrays are called out: ALP,SRM,SRK.
                     0043  5406 ;       For Subtest 2 these gate arrays are called out: SRM,SRK.
                     0043  5407 ;
                     0043  5408 ;       On error both subtests type out the following data:
                     0043  5409 ;
                     0043  5410 ;       EXPECTED WBUS DATA
                     0043  5411 ;       RECEIVED WBUS DATA
                     0043  5412 ;       LOOP COUNT I (PLATCH VALUE BEING USED)
                     0043  5413 ;       LOOP COUNT J (TEST PATTERN BEING USED)
                     0043  5414 ;       INDEX VALUE K (INDEX INTO EXPECTED DATA TABLE)
                     0043  5415 ;--
                     0043  5416 ;*******************************************************************************
                     0043  5417
                     0043  5418         SUBTEST         1
                     0046
                     0046       ;////////////////////////////////////////////////////////////////////////////
                     0046  5419 ;+
                     0046  5420 ; This subtest examines the second level shifter inside the ALP gate arrays.
                     0046  5421 ;-
                     0046  5422         INITIALIZE                                      ; INITIALIZE THE CPU
                     0048  5423         LDINDEX         K,1                             ; LOAD K WITH A 1
                     004F  5424         LOOP            I,1,4                           ; LOOP ON I FOR 4 PLATCH VALUES
                     0056  5425         LDFIELD         3$,I,W,10$,31,39                ; LITRL FIELD GETS PLATCH VALUE
                     0067  5426         LOADDCS         10$,,1,1                        ; LOAD PLATCH [LITRL] INTO DCS
                     006E  5427         LOADDCS         11$,,3,1                        ; SOURCE MTEMP0 AND RTEMP0
                     0075  5428         LOOP            J,1,6                           ; LOOP ON J FOR 6 TEST PATTERNS
                     007C  5429         SA01PAT         J,1$                            ; GENERATE A SA01 TEST PATTERN
                     0084  5430         LOADREG         WDREG,1$,,L                     ; LOAD PATTERN INTO WDREG
                     008C  5431         ERRLOOP                                         ; LOOP ON ERROR FROM HERE
```

```
                     008E  5432          FETCH           0                       ; EXECUTE THE ROT/RL.RM.P AND
                     0093  5433          BRSTCLK         4                       ; WRITE THE RESULT TO WHREG
                     0097  5434          CMPREG          WHREG,2$,K,L            ; IS WHREG CORRECT?
                     00A0  5435          IFERROR         ,<<ALP><SRM><SRK>>      ; IF NOT, SIGNAL AN ERROR
                     00A8  5436          INCINDEX        K                       ; INCREMENT K FOR NEXT PATTERN
                     00AB  5437          ENDLOOP         J                       ; CONTINUE J FOR ALL PATTERNS
                     00AE  5438          LOOP            J,7,7                   ; LOOP ONCE ON J (7TH PATTERN)
                     00B5  5439          LOADDCS         12$,,3,1                ; SOURCE MTEMP1 AND RTEMP2
                     00BC  5440          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                     00BE  5441          FETCH           0                       ; EXECUTE THE ROT/RL.RM.P AND
                     00C3  5442          BRSTCLK         4                       ; WRITE THE RESULT TO WHREG
                     00C7  5443          CMPREG          WHREG,2$,K,L            ; IS WHREG CORRECT?
                     00D0  5444          IFERROR         ,<<ALP><SRM><SRK>>      ; IF NOT, SIGNAL AN ERROR
                     00D8  5445          INCINDEX        K                       ; INCREMENT K FOR NEXT PATTERN
                     00DB  5446          ENDLOOP         J                       ; END LOOP ON J
                     00DE  5447          ENDLOOP         I                       ; CONTINUE I FOR ALL VALUES
                     00E1  5448
                     00E1  5449          SUBTEST         2
                     00E4
                     00E4              ;////////////////////////////////////////////////////////////////////////////
                     00E4  5450 ;+
                     00E4  5451 ; This subtest examines the first level shifter inside the SRM chips.
                     00E4  5452 ;-
                     00E4  5453          LDINDEX         K,1                     ; INITIALIZE INDEX K WITH 1
                     00EB  5454          LOOP            I,1,7                   ; LOOP ON I FOR 7 PLATCH VALUES
                     00F2  5455          LDFIELD         5$,I,W,10$,31,39        ; LITRL FIELD GETS A VALUE
                     0103  5456          LOADDCS         10$,,1,1                ; LOAD PLATCH_[LITRL] INTO DCS
                     010A  5457          LOADDCS         11$,,3,1                ; SOURCE MTEMP0 AND RTEMP0
                     0111  5458          LOOP            J,1,6                   ; LOOP ON J FOR 6 TEST PATTERNS
                     0118  5459          SA01PAT         J,1$                    ; GENERATE A SA01 TEST PATTERN
                     0120  5460          LOADREG         WDREG,1$,,L             ; LOAD PATTERN INTO WDREG
                     0128  5461          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                     012A  5462          FETCH           0                       ; EXECUTE THE ROT/RL.RM.P AND
                     012F  5463          BRSTCLK         4                       ; WRITE THE RESULT INTO WHREG
                     0133  5464          CMPREG          WHREG,4$,K,L            ; IS WHREG CORRECT?
                     013C  5465          IFERROR         ,<<SRM><SRK>>           ; IF NOT, SIGNAL AN ERROR
                     0143  5466          INCINDEX        K                       ; INCREMENT K FOR NEXT PATTERN
                     0146  5467          ENDLOOP         J                       ; CONTINUE J FOR ALL PATTERNS
                     0149  5468          LOOP            J,7,7                   ; LOOP ONCE ON J (7TH PATTERN)
                     0150  5469          LOADDCS         12$,,3,1                ; SOURCE MTEMP1 AND RTEMP2
                     0157  5470          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                     0159  5471          FETCH           0                       ; EXECUTE THE ROT/RL.RM.P AND
                     015E  5472          BRSTCLK         4                       ; WRITE THE RESULT INTO WHREG
                     0162  5473          CMPREG          WHREG,4$,K,L            ; IS WHREG CORRECT?
                     016B  5474          IFERROR         ,<<SRM><SRK>>           ; IF NOT, SIGNAL AN ERROR
                     0172  5475          INCINDEX        K                       ; INCREMENT K FOR NEXT PATTERN
                     0175  5476          ENDLOOP         J                       ; END LOOP ON J
                     0178  5477          ENDLOOP         I                       ; CONTINUE I FOR ALL VALUES
                     017B  5478          SKIP                                    ; SKIP TO NEXT TEST
                     0182  5479
                     0182  5480 BEGIN_TEST_DATA
                     0182
                     0182              ;-------------------------------------------------------------------------
                     0182  5481
00000000             0182  5482 1$:      .LONG   0       ; LOCATION TO HOLD SA01PAT FOR WDREG
```

```
                                0186  5483 ;
                                0186  5484 ; Expected data for Subtest 1
                                0186  5485 ;
                                0186  5486 ; The numbers to the right indicate the PLATCH and K index values.
                                0186  5487 ;
        33333333 55555555 AAAAAAAA  0186  5488 2$:     .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 0   K = 01-03
        0000FFFF 00FF00FF 0F0F0F0F  0192  5489         .LONG   ^X0F0F0F0F,^X00FF00FF,^X0000FFFF ;                K = 04-06
                          FFFFFFFF  019E  5490         .LONG   ^XFFFFFFFF                       ;                K = 07
        66666666 AAAAAAAA 55555555  01A2  5491         .LONG   ^X55555555,^XAAAAAAAA,^X66666666 ; PLATCH = 1   K = 08-0A
        0001FFFE 01FE01FE 1E1E1E1E  01AE  5492         .LONG   ^X1E1E1E1E,^X01FE01FE,^X0001FFFE ;                K = 0B-0D
                          00000001  01BA  5493         .LONG   ^X00000001                       ;                K = 0E
        CCCCCCCC 55555555 AAAAAAAA  01BE  5494         .LONG   ^XAAAAAAAA,^X55555555,^XCCCCCCCC ; PLATCH = 2   K = 0F-11
        0003FFFC 03FC03FC 3C3C3C3C  01CA  5495         .LONG   ^X3C3C3C3C,^X03FC03FC,^X0003FFFC ;                K = 12-14
                          00000003  01D6  5496         .LONG   ^X00000003                       ;                K = 15
        99999999 AAAAAAAA 55555555  01DA  5497         .LONG   ^X55555555,^XAAAAAAAA,^X99999999 ; PLATCH = 3   K = 16-18
        0007FFF8 07F807F8 78787878  01E6  5498         .LONG   ^X78787878,^X07F807F8,^X0007FFF8 ;                K = 19-1B
                          00000007  01F2  5499         .LONG   ^X00000007                       ;                K = 1C
                                01F6  5500
                                01F6  5501 ; PLATCH values for Subtest 1
                                01F6  5502
                           0000 01F6  5503 3$:     .WORD   0       ; I = 1
                           0001 01F8  5504         .WORD   1       ; I = 2
                           0002 01FA  5505         .WORD   2       ; I = 3
                           0003 01FC  5506         .WORD   3       ; I = 4
                                01FE  5507 ;
                                01FE  5508 ; Expected data for Subtest 2
                                01FE  5509 ;
                                01FE  5510 ; The numbers to the right indicate the PLATCH and K index values.
                                01FE  5511 ;
        33333333 55555555 AAAAAAAA  01FE  5512 4$:     .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 4   K = 01-03
        000FFFF0 0FF00FF0 F0F0F0F0  020A  5513         .LONG   ^XF0F0F0F0,^X0FF00FF0,^X000FFFF0 ;                K = 04-06
                          0000000F  0216  5514         .LONG   ^X0000000F                       ;                K = 07
        33333333 55555555 AAAAAAAA  021A  5515         .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 8   K = 08-0A
        00FFFF00 FF00FF00 0F0F0F0F  0226  5516         .LONG   ^X0F0F0F0F,^XFF00FF00,^X00FFFF00 ;                K = 0B-0D
                          000000FF  0232  5517         .LONG   ^X000000FF                       ;                K = 0E
        33333333 55555555 AAAAAAAA  0236  5518         .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 12  K = 0F-11
        0FFFF000 F00FF00F F0F0F0F0  0242  5519         .LONG   ^XF0F0F0F0,^XF00FF00F,^X0FFFF000 ;                K = 12-14
                          00000FFF  024E  5520         .LONG   ^X00000FFF                       ;                K = 15
        33333333 55555555 AAAAAAAA  0252  5521         .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 16  K = 16-18
        FFFF0000 00FF00FF 0F0F0F0F  025E  5522         .LONG   ^X0F0F0F0F,^X00FF00FF,^XFFFF0000 ;                K = 19-1B
                          0000FFFF  026A  5523         .LONG   ^X0000FFFF                       ;                K = 1C
        33333333 55555555 AAAAAAAA  026E  5524         .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 20  K = 1D-1F
        FFF0000F 0FF00FF0 F0F0F0F0  027A  5525         .LONG   ^XF0F0F0F0,^X0FF00FF0,^XFFF0000F ;                K = 20-22
                          000FFFFF  0286  5526         .LONG   ^X000FFFFF                       ;                K = 23
        33333333 55555555 AAAAAAAA  028A  5527         .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 24  K = 24-26
        FF0000FF FF00FF00 0F0F0F0F  0296  5528         .LONG   ^X0F0F0F0F,^XFF00FF00,^XFF0000FF ;                K = 27-29
                          00FFFFFF  02A2  5529         .LONG   ^X00FFFFFF                       ;                K = 2A
        33333333 55555555 AAAAAAAA  02A6  5530         .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; PLATCH = 28  K = 2B-2D
        F0000FFF F00FF00F F0F0F0F0  02B2  5531         .LONG   ^XF0F0F0F0,^XF00FF00F,^XF0000FFF ;                K = 2E-30
                          0FFFFFFF  02BE  5532         .LONG   ^X0FFFFFFF                       ;                K = 31
                                02C2  5533
                                02C2  5534 ; PLATCH values for Subtest 2
                                02C2  5535
                           0004 02C2  5536 5$:     .WORD   4       ; I = 1
                           0008 02C4  5537         .WORD   8       ; I = 2
```

```
                          000C  02C6  5538          .WORD   12      ; I = 3
                          0010  02C8  5539          .WORD   16      ; I = 4
                          0014  02CA  5540          .WORD   20      ; I = 5
                          0018  02CC  5541          .WORD   24      ; I = 6
                          001C  02CE  5542          .WORD   28      ; I = 7
                                02D0  5543
                                02D0  5544 ; Microword to load the PLATCH
                                02D0  5545
                                02D0  5546 10$:
U 01, 7700,1800,EF64,0300,0470,1802   02D0  5547 ;% 01   PL_[0]  ; LOADS THE PLATCH WITH THE CONTENTS OF THE LITRL FIELD
                                02DB  5551
                                02DB  5552 ; Microword to test six SA01 patterns
                                02DB  5553
                                02DB  5554 11$:
U 03, 6700,C800,8F47,0300,0470,1804   02DB  5555 ;% 03   MSRC/TEMP0,RSRC/TEMP0,ROT/RL.RM.P,ALPCTL/WX_S.Q_0,RDM_WB
                                02E6  5559
                                02E6  5560 ; Microword to test seventh SA01 pattern
                                02E6  5561
                                02E6  5562 12$:
U 03, 6700,C801,8F47,0308,0470,1804   02E6  5563 ;% 03   MSRC/TEMP1,RSRC/TEMP2,ROT/RL.RM.P,ALPCTL/WX_S.Q_0,RDM_WB
                                02F1  5567
                                02F1  5568 BEGIN_CPU_DATA
                                0002  5569
                                0002  5570 DCS_DATA
                                0001  5571
U 00, 7700,C800,0364,0300,0470,1801   0001  5572 ;% 00   NOP
U 01, 7700,1800,EF64,0300,0470,1802   000C  5576 ;% 01   PL_[0]
U 02, 5700,0840,0364,0300,0470,1803   0017  5580 ;% 02   R[TEMP0]_RDM
U 03, 6700,C800,8F47,0300,0470,1804   0022  5584 ;% 03   MSRC/TEMP0,RSRC/TEMP0,ROT/RL.RM.P,ALPCTL/WX_S.Q_0,RDM_WB
U 04, 7300,4800,0364,0300,0470,1805   002D  5588 ;% 04   STOP.CLOCK
                                0038  5592
                                0038  5593 RTEMP_DATA
                                0001  5594
                     00000000   0001  5595          .LONG   0               ; INITIALIZES RTEMP0 AND MTEMP0
                     00000000   0005  5596          .LONG   0               ; CONSTANT 0 FOR MTEMP1
                     FFFFFFFF   0009  5597          .LONG   ^XFFFFFFFF      ; CONSTANT 1 FOR RTEMP2
                                000D  5598
                                000D  5599 END_CPU_DATA
                                02F1  5600
                                02F1  5601 END_TEST_DATA
                                02F1
                                02F1       ;-----------------------------------------------------------------------------
                                02F1  5602
                                02F1  5603 ENDTEST
```

```
0043        .SBTTL  "TEST   053     Rotator Test 33 Left Rotation of RBUS concat. with MBUS Test
0043  5605 ;*******************************************************************************
0043  5606 ;++
0043  5607 ;
0043  5608 ; FUNCTIONAL DESCRIPTION
0043  5609 ;
0043  5610 ;       This test examines the two rotator micro-orders that rotate the RBUS
0043  5611 ;       concatenated with the MBUS left by four bits and by the sum of the
0043  5612 ;       contents of the SLATCH and PLATCH. (ROT/RL.RM.4 and ROT/RL.RM.PS)
0043  5613 ;
0043  5614 ; TEST ALGORITHM
0043  5615 ;
0043  5616 ;       This test is implemented in three subtests. Subtest 1 examines the left
0043  5617 ;       rotation of RBUS and MBUS by four bits. Subtest 2 uses a floating zero
0043  5618 ;       pattern in the SLATCH and a constant one in the PLATCH to test the left
0043  5619 ;       rotation of RBUS and MBUS by the sum of the two latches. Subtest 3 uses
0043  5620 ;       a floating zero pattern in the PLATCH and a constant one in the SLATCH
0043  5621 ;       to test the left rotation of RBUS and MBUS by the sum of the latches.
0043  5622 ;       The algorithm for Subtest 1 and Subtests 2 and 3 are as follows:
0043  5623 ;
0043  5624 ;       Subtest 1
0043  5625 ;
0043  5626 ;        1. Initialize the CPU.
0043  5627 ;        2. Load DCS with the micro-instruction ROT/RL.RM.4 using the contents
0043  5628 ;           of scratchpad TEMP0 to drive both busses.
0043  5629 ;        3. Start signature analysis.
0043  5630 ;        4. Loop on I for 6 stuck at zero or one test patterns.
0043  5631 ;        5. Generate a stuck at zero or one test pattern based on I.
0043  5632 ;        6. Load the test pattern into WDREG.
0043  5633 ;        7. Loop on error from here.
0043  5634 ;        8. Execute the micro-code to:
0043  5635 ;               1. Load the PLATCH with the contents of the LITRL field.
0043  5636 ;               2. Load the SLATCH with the contents of the LITRL field.
0043  5637 ;               3. Load RTEMP0 with the contents of WDREG.
0043  5638 ;               4. Execute the ROT/RL.RM.4 and write the result to WHREG.
0043  5639 ;        9. Does the WHREG match the expected data from a table indexed by I?
0043  5640 ;       10. If not, signal an error.
0043  5641 ;       11. Continue the loop on I for the remaining test patterns.
0043  5642 ;       12. Loop once on I equal to 7.
0043  5643 ;       13. Load DCS with the micro-instruction ROT/RL.RM.4 using the contents
0043  5644 ;           of scratchpads MTEMP1 and RTEMP2 to drive the two busses.
0043  5645 ;       14. Loop on error from here.
0043  5646 ;       15. Run the micro-code for the rotation and write the result to WHREG.
0043  5647 ;       16. Does the WHREG match the expected data from a table indexed by I?
0043  5648 ;       17. If not, signal an error.
0043  5649 ;       18. End the single loop on I equal to seven.
0043  5650 ;       19. Stop signature analysis.
0043  5651 ;
0043  5652 ;       Subtests 2 and 3
0043  5653 ;
0043  5654 ;        1. Load the LITRL field of a PLATCH gets LITRL (Subtest 2) or SLATCH
0043  5655 ;           gets LITRL (Subtest 3) micro-instruction with a constant 1.
0043  5656 ;        2. Load the instruction into DCS.
0043  5657 ;        3. Initialize the index K with a 1.
0043  5658 ;        4. Start signature analysis.
```

```
0043  5659 ;        5. Loop on I for 5 SLATCH (Subtest 2) or PLATCH (Subtest 3) values.
0043  5660 ;        6. Load the LITRL field of a SLATCH gets LITRL (Subtest 2) or PLATCH
0043  5661 ;           gets LITRL (Subtest 3) micro-instruction with a floating zero from
0043  5662 ;           a table indexed by I.
0043  5663 ;        7. Load the instruction into DCS.
0043  5664 ;        8. Load a ROT/RL.RM.PS test instruction with the source fields set to
0043  5665 ;           source MTEMP0 and RTEMP0.
0043  5666 ;        9. Loop on J for 6 stuck at zero or one test patterns.
0043  5667 ;       10. Generate a stuck at zero or one test pattern based on J.
0043  5668 ;       11. Load the WDREG with the test pattern.
0043  5669 ;       12. Loop on error from here.
0043  5670 ;       13. Execute the micro-code to:
0043  5671 ;                1. Load the PLATCH with the contents of the LITRL field.
0043  5672 ;                2. Load the SLATCH with the contents of the LIRTL field.
0043  5673 ;                3. Load RTEMP0 with the contents of the WDREG.
0043  5674 ;                4. Using the contents of TEMP0 to drive both busses, execute a
0043  5675 ;                   left rotation of the RBUS and MBUS by the sum of SLATCH and
0043  5676 ;                   PLATCH. Write the result to the WHREG.
0043  5677 ;       14. Does the WHREG match the expected data from a table indexed by K?
0043  5678 ;       15. If not, signal an error.
0043  5679 ;       16. Increment the index K for the next test pattern.
0043  5680 ;       17. Continue the loop on J for the remaining test patterns.
0043  5681 ;       18. Loop once on J equal to 7.
0043  5682 ;       19. With the source fields set to source MTEMP2 and RTEMP1, load a
0043  5683 ;           ROT/RL.RM.PS test instruction into DCS.
0043  5684 ;       20. Loop on error from here.
0043  5685 ;       21. Run the rotation and write the result into the WHREG.
0043  5686 ;       22. Does the WHREG match the expected data from a table indexed by K?
0043  5687 ;       23. If not, signal an error.
0043  5688 ;       24. Increment the index K for the next test pattern.
0043  5689 ;       25. End the loop on J equal to 7.
0043  5690 ;       26. Continue the loop on I for the remaining SLATCH (Subtest 2) or
0043  5691 ;           PLATCH (Subtest 3) values.
0043  5692 ;       27. Stop signature analysis.
0043  5693 ;       28. End of this test, skip to the next test. (Subtest 3 only)
0043  5694 ;
0043  5695 ; TEST DATA
0043  5696 ;
0043  5697 ;       Because RBUS concatenated with MBUS is 64 bits wide, seven stuck at
0043  5698 ;       zero or one test patterns must be used to check the rotation logic.
0043  5699 ;       The first six patterns contain the same data for both busses and can be
0043  5700 ;       placed into a single longword using the SA01PAT psuedo-op. The seventh
0043  5701 ;       pattern places all zeros on the MBUS and all ones on the RBUS. These
0043  5702 ;       two longwords are stored into the scratchpads MTEMP1 and RTEMP2, which
0043  5703 ;       are driven onto the MBUS and RBUS at the appropriate time. The number
0043  5704 ;       of bit positions each test pattern is rotated by depends on the subtest
0043  5705 ;       under execution. The subtest dependent data is as follows:
0043  5706 ;
0043  5707 ;       Subtest 1
0043  5708 ;
0043  5709 ;       The expected data for this subtest consists of the seven stuck at zero
0043  5710 ;       or one test patterns rotated left by four bits.
0043  5711 ;
0043  5712 ;       Subtests 2 and 3
0043  5713 ;
```

```
                        0043  5714 ;       The expected data for these two subtests consists of the seven stuck at
                        0043  5715 ;       zero or one test patterns rotated left by 31, 30, 28, 24 and 16 bits.
                        0043  5716 ;       These numbers are obtained when a one is added to a 5 bit floating zero
                        0043  5717 ;       pattern. The floating zero patterns are used to check the logic that
                        0043  5718 ;       adds the SLATCH and PLATCH.
                        0043  5719 ;
                        0043  5720 ; LOGIC DESCRIPTION
                        0043  5721 ;
                        0043  5722 ;       The logic involved in this test is contained in the SRM, SRK, and ALP
                        0043  5723 ;       gate arrays. SRK decodes the ROT field of the micro-word to generate
                        0043  5724 ;       the control signals, SHF<4:0>, PRI<1:0>, and SEC<5:0> that control the
                        0043  5725 ;       rotation logic inside the SRM and ALP slices.
                        0043  5726 ;
                        0043  5727 ; ASSUMPTIONS
                        0043  5728 ;
                        0043  5729 ;       This test assumes the following logic has been tested and is working;
                        0043  5730 ;       the MBUS, RBUS, SBUS and WBUS; the dual port scratchpads TEMP0, TEMP1
                        0043  5731 ;       and TEMP2; the PLATCH and SLATCH; and the WMUX_S.Q_0 ALPCTL data path.
                        0043  5732 ;
                        0043  5733 ; ERROR DESCRIPTION
                        0043  5734 ;
                        0043  5735 ;       For Subtest 1 these gate arrays are called out: SRK,SRM.
                        0043  5736 ;       For Subtests 2 and 3 these gate arrays are called out: SRK,SRM,ALP.
                        0043  5737 ;
                        0043  5738 ;       On error Subtest 1 types out the following data:
                        0043  5739 ;
                        0043  5740 ;       EXPECTED WBUS DATA
                        0043  5741 ;       RECEIVED WBUS DATA
                        0043  5742 ;       LOOP COUNT I (TEST PATTERN BEING USED)
                        0043  5743 ;
                        0043  5744 ;       On error Subtests 2 and 3 type out the following data:
                        0043  5745 ;
                        0043  5746 ;       EXPECTED WBUS DATA
                        0043  5747 ;       RECEIVED WBUS DATA
                        0043  5748 ;       LOOP COUNT I (PLATCH VALUE BEING USED)
                        0043  5749 ;       LOOP COUNT J (TEST PATTERN BEING USED)
                        0043  5750 ;       INDEX VALUE K (INDEX INTO EXPECTED DATA TABLE)
                        0043  5751 ;--
                        0043  5752 ;*******************************************************************************
                        0043  5753
                        0043  5754         SUBTEST         1
                        0046
                        0046       ;////////////////////////////////////////////////////////////////////////////
                        0046  5755 ;+
                        0046  5756 ; This subtest examines the ROT/RL.RM.4 micro-order.
                        0046  5757 ;-
                        0046  5758         INITIALIZE                              ; INITIALIZE THE CPU
                        0048  5759         LOADDCS         14$,,4,1                ; SOURCE MTEMP0 AND RTEMP0
                        004F  5760         LOOP            I,1,6                   ; LOOP FOR 6 SA01 TEST PATTERNS
                        0056  5761         SA01PAT         I,1$                    ; GENERATE A SA01 TEST PATTERN
                        005E  5762         LOADREG         WDREG,1$,,L             ; LOAD THE PATTERN INTO WDREG
                        0066  5763         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        0068  5764         FETCH           0                       ; ROT/RL.RM.4 WITH MTEMP0 AND
                        006D  5765         BRSTCLK         5                       ; RTEMP0 WRITE RESULT TO WHREG
                        0071  5766         CMPREG          WHREG,6$,I,L            ; IS WHREG CORRECT?
```

```
                007A  5767          IFERROR         ,<<SRK><SRM>>                   ; IF NOT, SIGNAL AN ERROR
                0081  5768          ENDLOOP         I                               ; LOOP FOR THE OTHER PATTERNS
                0084  5769          LOOP            I,7,7                           ; LOOP ONCE ON I = 7
                008B  5770          LOADDCS         15$,,4,1                        ; SOURCE MTEMP1 AND RTEMP2
                0092  5771          ERRLOOP                                         ; LOOP ON ERROR FROM HERE
                0094  5772          FETCH           0                               ; ROT/RL.RM.4 WITH MTEMP1 AND
                0099  5773          BRSTCLK         5                               ; RTEMP2 WRITE RESULT TO WHREG
                009D  5774          CMPREG          WHREG,6$,I,L                    ; IS WHREG CORRECT?
                00A6  5775          IFERROR         ,<<SRK><SRM>>                   ; IF NOT, SIGNAL AN ERROR
                00AD  5776          ENDLOOP         I                               ; END LOOP ON I = 7
                00B0  5777
                00B0  5778          SUBTEST         2
                00B3
                00B3        ;////////////////////////////////////////////////////////////////////////////////
                00B3  5779  ;+
                00B3  5780  ; This subtest examines the ROT/RL.RM.PS micro-order using a floating zero
                00B3  5781  ; pattern in the SLATCH with a constant one in the PLATCH.
                00B3  5782  ;-
                00B3  5783          LDFIELD         9$,,W,10$,31,39                 ; LITRL FIELD GETS A 1
                00C4  5784          LOADDCS         10$,,1,1                        ; LOAD PLATCH_[1] INTO DCS
                00CB  5785          LDINDEX         K,1                             ; INITIALIZE INDEX K
                00D2  5786          LOOP            I,1,5                           ; LOOP ON I FOR 5 SLATCH VALUES
                00D9  5787          LDFIELD         8$,I,W,11$,31,39                ; LITRL FIELD GETS A FLOATING 0
                00EA  5788          LOADDCS         11$,,2,1                        ; LOAD SLATCH_[] INST. INTO DCS
                00F1  5789          LOADDCS         16$,,4,1                        ; MSRC/TEMP0 AND RSRC/TEMP0
                00F8  5790          LOOP            J,1,6                           ; LOOP ON J FOR 6 SA01 PATTERNS
                00FF  5791          SA01PAT         J,1$                            ; GENERATE A SA01 PATTERN
                0107  5792          LOADREG         WDREG,1$,,L                     ; LOAD THE PATTERN INTO WDREG
                010F  5793          ERRLOOP                                         ; LOOP ON ERROR FROM HERE
                0111  5794          FETCH           0                               ; ROT/RL.RM.PS USING MTEMP0 AND
                0116  5795          BRSTCLK         5                               ; RTEMP0 WRITE RESULT TO WHREG
                011A  5796          CMPREG          WHREG,7$,K,L                    ; IS WHREG CORRECT?
                0123  5797          IFERROR         ,<<SRK><SRM><ALP>>              ; IF NOT, SIGNAL AN ERROR
                012B  5798          INCINDEX        K                               ; INCREMENT K FOR NEXT PATTERN
                012E  5799          ENDLOOP         J                               ; CONTINUE J FOR OTHER PATTERNS
                0131  5800          LOOP            J,7,7                           ; LOOP ONCE ON J = 7
                0138  5801          LOADDCS         17$,,4,1                        ; MSRC/TEMP1 AND RSRC/TEMP2
                013F  5802          ERRLOOP                                         ; LOOP ON ERROR FROM HERE
                0141  5803          FETCH           0                               ; ROT/RL.RM.PS USING MTEMP1 AND
                0146  5804          BRSTCLK         5                               ; RTEMP2 WRITE RESULT TO WHREG
                014A  5805          CMPREG          WHREG,7$,K,L                    ; IS WHREG CORRECT?
                0153  5806          IFERROR         ,<<SRK><SRM><ALP>>              ; IF NOT, SIGNAL AN ERROR
                015B  5807          INCINDEX        K                               ; INCREMENT K FOR NEXT PATTERN
                015E  5808          ENDLOOP         J                               ; END OF LOOP ON J = 7
                0161  5809          ENDLOOP         I                               ; CONTINUE I FOR OTHER FLOATS
                0164  5810
                0164  5811          SUBTEST         3
                0167
                0167        ;////////////////////////////////////////////////////////////////////////////////
                0167  5812  ;+
                0167  5813  ; This subtest examines the ROT/RL.RM.PS micro-order using a floating zero
                0167  5814  ; pattern in the PLATCH with a constant one in the SLATCH.
                0167  5815  ;-
                0167  5816          LDFIELD         9$,,W,11$,31,39                 ; LITRL FIELD GETS A 1
                0178  5817          LOADDCS         11$,,2,1                        ; LOAD SLATCH_[1] INTO DCS
```

```
                        017F  5818          LDINDEX         K,1                             ; INITIALIZE INDEX K WITH 1
                        0186  5819          LOOP            I,1,5                           ; LOOP ON I FOR 5 PLATCH VALUES
                        018D  5820          LDFIELD         8$,I,W,10$,31,39                ; LITRL FIELD GETS A FLOATING 0
                        019E  5821          LOADDCS         10$,,1,1                        ; LOAD PLATCH_[] INST. INTO DCS
                        01A5  5822          LOADDCS         16$,,4,1                        ; MSRC/TEMP0 AND RSRC/TEMP0
                        01AC  5823          LOOP            J,1,6                           ; LOOP ON J FOR 6 SA01 PATTERNS
                        01B3  5824          SA01PAT         J,1$                            ; GENERATE A SA01 TEST PATTERN
                        01BB  5825          LOADREG         WDREG,1$,,L                     ; LOAD PATTERN INTO WDREG
                        01C3  5826          ERRLOOP                                         ; LOOP ON ERROR FROM HERE
                        01C5  5827          FETCH           0                               ; ROT/RL.RM.PS USING MTEMP0 AND
                        01CA  5828          BRSTCLK         5                               ; RTEMP0 WRITE RESULT TO WHREG
                        01CE  5829          CMPREG          WHREG,7$,K,L                    ; IS WHREG CORRECT?
                        01D7  5830          IFERROR         ,<<SRK><SRM><ALP>>              ; IF NOT, SIGNAL AN ERROR
                        01DF  5831          INCINDEX        K                               ; INCREMENT K FOR NEXT PATTERN
                        01E2  5832          ENDLOOP         J                               ; CONTINUE J FOR OTHER PATTERNS
                        01E5  5833          LOOP            J,7,7                           ; LOOP ONCE ON J = 7
                        01EC  5834          LOADDCS         17$,,4,1                        ; MSRC/TEMP1 AND RSRC/TEMP2
                        01F3  5835          ERRLOOP                                         ; LOOP ON ERROR FROM HERE
                        01F5  5836          FETCH           0                               ; ROT/RL.RM.PS USING MTEMP1 AND
                        01FA  5837          BRSTCLK         5                               ; RTEMP2 WRITE RESULT TO WHREG
                        01FE  5838          CMPREG          WHREG,7$,K,L                    ; IS WHREG CORRECT?
                        0207  5839          IFERROR         ,<<SRK><SRM><ALP>>              ; IF NOT, SIGNAL AN ERROR
                        020F  5840          INCINDEX        K                               ; INCREMENT K FOR NEXT PATTERN
                        0212  5841          ENDLOOP         J                               ; END THE LOOP ON J = 7
                        0215  5842          ENDLOOP         I                               ; CONTINUE I FOR OTHER FLOATS
                        0218  5843          SKIP                                            ; END OF TEST
                        021F  5844
                        021F  5845 BEGIN_TEST_DATA
                        021F
                        021F          ;-----------------------------------------------------------------------
                        021F  5846
                00000000 021F  5847 1$:     .LONG   0       ; LOCATION TO HOLD SA01 PATTERN TO LOAD WDREG
                        0223  5848
                        0223  5849 ; Expected data for Subtest 1
                        0223  5850 ;
                        0223  5851 ; This data consists of seven stuck at zero or one test patterns rotated left
                        0223  5852 ; by four bits.
                        0223  5853 ;
                        0223  5854
33333333 55555555 AAAAAAAA 0223  5855 6$:     .LONG   ^XAAAAAAAA,^X55555555,^X33333333        ; I = 1-3
000FFFF0 0FF00FF0 F0F0F0F0 022F  5856         .LONG   ^XF0F0F0F0,^X0FF00FF0,^X000FFFF0        ; I = 4-6
                0000000F 023B  5857         .LONG   ^X0000000F                              ; I = 7
                        023F  5858
                        023F  5859 ; Expected data for Subtests 2 and 3
                        023F  5860 ;
                        023F  5861 ; This data consists of seven stuck at zero or one test patterns rotated left
                        023F  5862 ; by 31, 30, 28, 24 and 16 bits. The numbers in the comments to the right
                        023F  5863 ; indicate the floating pattern being used and the K index values for each data
                        023F  5864 ; pattern. The floating patterns are loaded into the SLATCH for Subtest 2 and
                        023F  5865 ; into the PLATCH for Subtest 3. A one is loaded into the PLATCH for Subtest 2
                        023F  5866 ; and the SLATCH for Subtest 3.
                        023F  5867
99999999 AAAAAAAA 55555555 023F  5868 7$:     .LONG   ^X55555555,^XAAAAAAAA,^X99999999 ; FLOAT = 30   K = 01-03
80007FFF 807F807F 87878787 024B  5869         .LONG   ^X87878787,^X807F807F,^X80007FFF ;              K = 04-06
                7FFFFFFF 0257  5870         .LONG   ^X7FFFFFFF                       ;              K = 07
```

```
        CCCCCCCC 55555555 AAAAAAAA  025B  5871          .LONG   ^XAAAAAAAA,^X55555555,^XCCCCCCCC ; FLOAT = 29   K = 08-0A
        C0003FFF C03FC03F C3C3C3C3  0267  5872          .LONG   ^XC3C3C3C3,^XC03FC03F,^XC0003FFF ;                K = 0B-0D
                          3FFFFFFF  0273  5873          .LONG   ^X3FFFFFFF                       ;                K = 0E
        33333333 55555555 AAAAAAAA  0277  5874          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; FLOAT = 27   K = 0F-11
        F0000FFF F00FF00F F0F0F0F0  0283  5875          .LONG   ^XF0F0F0F0,^XF00FF00F,^XF0000FFF ;                K = 12-14
                          0FFFFFFF  028F  5876          .LONG   ^X0FFFFFFF                       ;                K = 15
        33333333 55555555 AAAAAAAA  0293  5877          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; FLOAT = 23   K = 16-18
        FF0000FF FF00FF00 0F0F0F0F  029F  5878          .LONG   ^X0F0F0F0F,^XFF00FF00,^XFF0000FF ;                K = 19-1B
                          00FFFFFF  02AB  5879          .LONG   ^X00FFFFFF                       ;                K = 1C
        33333333 55555555 AAAAAAAA  02AF  5880          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; FLOAT = 15   K = 1D-1F
        FFFF0000 00FF00FF 0F0F0F0F  02BB  5881          .LONG   ^X0F0F0F0F,^X00FF00FF,^XFFFF0000 ;                K = 20-22
                          0000FFFF  02C7  5882          .LONG   ^X0000FFFF                       ;                K = 23
                                    02CB  5883
                                    02CB  5884 ; SLATCH and PLATCH values for Subtests 2 and 3
                                    02CB  5885
          000F 0017 001B 001D 001E  02CB  5886 8$:      .WORD   ^X1E,^X1D,^X1B,^X17,^X0F         ; FLOATING ZERO PATTERNS
                               0001 02D5  5887 9$:      .WORD   ^X01                             ; ONE
                                    02D7  5888
                                    02D7  5889 ; Microwords to load the PLATCH and SLATCH
                                    02D7  5890
                                    02D7  5891 10$:
U 01, 7700,1800,EF64,0300,0470,1802 02D7  5892 ;% 01    PL_[0]  ; PLATCH GETS THE CONTENTS OF THE LITRL FIELD
                                    02E2  5896
                                    02E2  5897 11$:
U 02, 7700,9800,F764,0300,0470,1803 02E2  5898 ;% 02    SL_[0]  ; SLATCH GETS THE CONTENTS OF THE LITRL FIELD
                                    02ED  5902
                                    02ED  5903 ; Test microwords for Subtest 1
                                    02ED  5904 ;
                                    02ED  5905 ; This microword is used to test the left rotation of the RBUS concatenated
                                    02ED  5906 ; with the MBUS by four bits using six stuck at zero or one test patterns.
                                    02ED  5907
                                    02ED  5908 14$:
U 04, 6700,4800,2347,0300,0470,1805 02ED  5909 ;% 04    MSRC/TEMP0,RSRC/TEMP0,ROT/RL.RM.4,ALPCTL/WX_S.Q_0,RDM_WB
                                    02F8  5913
                                    02F8  5914 ; This microword is used to test the left rotation of the RBUS concatenated
                                    02F8  5915 ; with the MBUS by four bits using the seventh stuck at zero or one pattern.
                                    02F8  5916
                                    02F8  5917 15$:
U 04, 6700,4801,2347,0308,0470,1805 02F8  5918 ;% 04    MSRC/TEMP1,RSRC/TEMP2,ROT/RL.RM.4,ALPCTL/WX_S.Q_0,RDM_WB
                                    0303  5922
                                    0303  5923 ; Test microwords for Subtests 2 and 3
                                    0303  5924 ;
                                    0303  5925 ; This microword is used to test the left rotation of the RBUS concatenated
                                    0303  5926 ; with the MBUS. The busses are rotated by the number of bits given in the sum
                                    0303  5927 ; of the S and P latches. The input data is the first six stuck at zero or one
                                    0303  5928 ; test patterns.
                                    0303  5929
                                    0303  5930 16$:
U 04, 6700,4800,8347,0300,0470,1805 0303  5931 ;% 04    MSRC/TEMP0,RSRC/TEMP0,ROT/RL.RM.PS,ALPCTL/WX_S.Q_0,RDM_WB
                                    030E  5935
                                    030E  5936 ; This microword is used to test the left rotation of the RBUS concatenated
                                    030E  5937 ; with the MBUS. The busses are rotated by the number of bits given in the sum
                                    030E  5938 ; of the S and P latches. The input data is the seventh stuck at zero or one
                                    030E  5939 ; test pattern.
                                    030E  5940
```

```
                                 030E  5941 17$:
U 04, 6700,4801,8347,0308,0470,1805  030E  5942 ;% 04   MSRC/TEMP1,RSRC/TEMP2,ROT/RL.RM.PS,ALPCTL/WX_S.Q_0,RDM_WB
                                 0319  5946
                                 0319  5947 BEGIN_CPU_DATA
                                 0002  5948
                                 0002  5949 DCS_DATA
                                 0001  5950
U 00, 7700,C800,0364,0300,0470,1801  0001  5951 ;% 00   NOP
U 01, 7700,1800,EF64,0300,0470,1802  000C  5955 ;% 01   PL_[0]
U 02, 7700,9800,F764,0300,0470,1803  0017  5959 ;% 02   SL_[0]
U 03, 5700,8840,0364,0300,0470,1804  0022  5963 ;% 03   R[TEMP0]_RDM
U 04, 6700,4800,8F47,0300,0470,1805  002D  5967 ;% 04   MSRC/TEMP0,RSRC/TEMP0,ROT/RL.RM.P,ALPCTL/WX_S.Q_0,RDM_WB
U 05, 7300,4800,0364,0300,0470,1806  0038  5971 ;% 05   STOP.CLOCK
                                 0043  5975
                                 0043  5976 RTEMP_DATA
                                 0001  5977
                        00000000 0001  5978         .LONG   0               ; INITIALIZES RTEMP0 AND MTEMP0
                        00000000 0005  5979         .LONG   0               ; CONSTANT 0 FOR MTEMP1
                        FFFFFFFF 0009  5980         .LONG   ^XFFFFFFFF      ; CONSTANT 1 FOR RTEMP2
                                 000D  5981
                                 000D  5982 END_CPU_DATA
                                 0319  5983
                                 0319  5984 END_TEST_DATA
                                 0319
                                 0319       ;-----------------------------------------------------------------------
                                 0319  5985
                                 0319  5986 ENDTEST
```

```
003D        .SBTTL  "TEST   054     Rotator Test 34 Zero Extend and Rotate Left Literal Test
003D  5988 ;*******************************************************************************
003D  5989 ;++
003D  5990 ;
003D  5991 ; FUNCTIONAL DESCRIPTION
003D  5992 ;
003D  5993 ;       This test examines the rotator's ability to zero extend a nine bit
003D  5994 ;       literal and rotate it left by a specific number of bits.
003D  5995 ;
003D  5996 ; TEST ALGORITHM
003D  5997 ;
003D  5998 ;        1. Initialize the CPU.
003D  5999 ;        2. Load the index K with a 1.
003D  6000 ;        3. Start signature analysis.
003D  6001 ;        4. Loop on I for 7 ROT/ZLIT micro-orders.
003D  6002 ;        5. Load the ROT field of the test micro-instruction with the proper
003D  6003 ;           ZLIT micro-order as read from a table indexed by I.
003D  6004 ;        6. Loop on J for 5 stuck at zero or one test patterns.
003D  6005 ;        7. Generate a stuck at zero or one test pattern based on J.
003D  6006 ;        8. Load the LITRL field of the test micro-instr. with the pattern.
003D  6007 ;        9. Load the modified test micro-instruction into DCS.
003D  6008 ;       10. Loop on error from here.
003D  6009 ;       11. Execute the micro-code to:
003D  6010 ;               1. Zero extend and left rotate the test pattern in the LITRL
003D  6011 ;                  field by the number of bits specified by the ZLIT micro-
003D  6012 ;                  order in the ROT field. Write the result to WHREG.
003D  6013 ;       12. Does the WHREG match the expected data from a table indexed by K?
003D  6014 ;       13. If not, signal an error.
003D  6015 ;       14. Increment the index K for the next test pattern.
003D  6016 ;       15. Continue the loop on J for the remaining test patterns.
003D  6017 ;       16. Continue the loop on I for the remaining ROT field micro-orders.
003D  6018 ;       17. Stop signature analysis.
003D  6019 ;       18. End of this test, skip to the next test.
003D  6020 ;
003D  6021 ; TEST DATA
003D  6022 ;
003D  6023 ;       The expected data for this test consists of the five stuck at zero or
003D  6024 ;       one test patterns zero extended and left rotated the proper number of
003D  6025 ;       bits for each of the seven ZLIT micro-orders tested.
003D  6026 ;
003D  6027 ; LOGIC DESCRIPTION
003D  6028 ;
003D  6029 ;       The logic used to perform the zero extension and left rotation of the
003D  6030 ;       LITRL field is internal to the SRM slices. The SRK gate array decodes
003D  6031 ;       the ROT field of the micro-word to generate the control signals,
003D  6032 ;       SHF<4:0>, PRI<1:0>, and SEC<5:0> that control the SRM slices.
003D  6033 ;
003D  6034 ; ASSUMPTIONS
003D  6035 ;
003D  6036 ;       This test assumes the following logic has been tested and is working;
003D  6037 ;       the SBUS and WBUS and the WMUX_S.Q_0 ALPCTL data path.
003D  6038 ;
003D  6039 ; ERROR DESCRIPTION
003D  6040 ;
003D  6041 ;       On error this test will call out the SRM and SRK gate arrays.
```

```
                        003D  6042 ;
                        003D  6043 ;       The following data will also by typed:
                        003D  6044 ;
                        003D  6045 ;       EXPECTED WBUS DATA
                        003D  6046 ;       RECEIVED WBUS DATA
                        003D  6047 ;       THE NUMBER OF THE ZLIT MICRO-ORDER UNDER TEST (INDEX I)
                        003D  6048 ;       THE NUMBER OF THE TEST PATTERN BEING USED (INDEX J)
                        003D  6049 ;       THE INDEX INTO THE EXPECTED DATA TABLE (INDEX K)
                        003D  6050 ;
                        003D  6051 ;--
                        003D  6052 ;*******************************************************************************
                        003D  6053 ;///////////////////////////////////////////////////////////////////////////////
                        003D  6054
                        003D  6055         INITIALIZE                                      ; INITIALIZE THE CPU
                        003F  6056         LDINDEX         K,1                             ; LOAD INDEX K WITH A 1
                        0046  6057         LOOP            I,1,7                           ; LOOP ON I FOR 7 ZLIT ORDERS
                        004D  6058         LDFIELD         2$,I,B,20$,58,63                ; ROT FIELD GETS A ZLIT ORDER
                        005E  6059         LOOP            J,1,5                           ; LOOP ON J FOR 5 TEST PATTERNS
                        0065  6060         SA01PAT         J,1$                            ; GENERATE A SA01 TEST PATTERN
                        006D  6061         LDFIELD         1$,,W,20$,31,39                 ; LITRL FIELD GETS TEST PATTERN
                        007E  6062         LOADDCS         20$,,1,1                        ; LOAD TEST INSTRUCTION TO DCS
                        0085  6063         ERRLOOP                                         ; LOOP ON ERROR FROM HERE
                        0087  6064         FETCH           0                               ; EXECUTE THE ROT/ZLIT ORDER
                        008C  6065         BRSTCLK         2                               ; AND WRITE THE RESULT TO WHREG
                        0090  6066         CMPREG          WHREG,10$,K,L                   ; IS WHREG CORRECT?
                        0099  6067         IFERROR         ,<<SRM><SRK>>                   ; IF NOT, SIGNAL AN ERROR
                        00A0  6068         INCINDEX        K                               ; INCREMENT K FOR NEXT PATTERN
                        00A3  6069         ENDLOOP         J                               ; CONTINUE J FOR OTHER PATTERNS
                        00A6  6070         ENDLOOP         I                               ; CONTINUE I FOR REST OF ZLITS
                        00A9  6071         SKIP                                            ; END OF TEST, SKIP TO NEXT
                        00B0  6072
                        00B0  6073 BEGIN_TEST_DATA
                        00B0
                        00B0       ;-------------------------------------------------------------------------
                        00B0  6074
                00000000 00B0  6075 1$:     .LONG   0       ; LOCATION TO HOLD THE SA01 PATTERN FOR WDREG
                        00B4  6076
                        00B4  6077 ; Table of ZLIT micro-orders.
                        00B4  6078
                     37 00B4  6079 2$:      .BYTE   ^X37    ; ZLIT4         I = 1
                     36 00B5  6080          .BYTE   ^X36    ; ZLIT8         I = 2
                     35 00B6  6081          .BYTE   ^X35    ; ZLIT12        I = 3
                     34 00B7  6082          .BYTE   ^X34    ; ZLIT16        I = 4
                     33 00B8  6083          .BYTE   ^X33    ; ZLIT20        I = 5
                     32 00B9  6084          .BYTE   ^X32    ; ZLIT24        I = 6
                     31 00BA  6085          .BYTE   ^X31    ; ZLIT28        I = 7
                        00BB  6086
                        00BB  6087 ; Expected Data
                        00BB  6088
00001330 00001550 00000AA0 00BB  6089 10$:     .LONG   ^X00000AA0,^X00001550,^X00001330 ; ROT/ZLIT4    K = 01-03
         00000FF0 000010F0 00C7  6090          .LONG   ^X000010F0,^X00000FF0            ;               K = 04-05
00013300 00015500 0000AA00 00CF  6091          .LONG   ^X0000AA00,^X00015500,^X00013300 ; ROT/ZLIT8    K = 06-08
         0000FF00 00010F00 00DB  6092          .LONG   ^X00010F00,^X0000FF00            ;               K = 09-0A
00133000 00155000 000AA000 00E3  6093          .LONG   ^X000AA000,^X00155000,^X00133000 ; ROT/ZLIT12   K = 0B-0D
         000FF000 0010F000 00EF  6094          .LONG   ^X0010F000,^X000FF000            ;               K = 0E-0F
```

```
         01330000 01550000 00AA0000  00F7  6095          .LONG   ^X00AA0000,^X01550000,^X01330000 ; ROT/ZLIT16   K = 10-12
                  00FF0000 010F0000  0103  6096          .LONG   ^X010F0000,^X00FF0000            ;               K = 13-14
         13300000 15500000 0AA00000  010B  6097          .LONG   ^X0AA00000,^X15500000,^X13300000 ; ROT/ZLIT20   K = 15-17
                  0FF00000 10F00000  0117  6098          .LONG   ^X10F00000,^X0FF00000            ;               K = 18-19
         33000001 55000001 AA000000  011F  6099          .LONG   ^XAA000000,^X55000001,^X33000001 ; ROT/ZLIT24   K = 1A-1C
                  FF000000 0F000001  012B  6100          .LONG   ^X0F000001,^XFF000000            ;               K = 1D-1E
         30000013 50000015 A000000A  0133  6101          .LONG   ^XA000000A,^X50000015,^X30000013 ; ROT/ZLIT28   K = 1F-21
                  F000000F F0000010  013F  6102          .LONG   ^XF0000010,^XF000000F            ;               K = 22-23
                                     0147  6103
                                     0147  6104 ; Test micro-instruction
                                     0147  6105
                                     0147  6106 20$:
U 01, 6700,5800,DF47,0355,0470,1802  0147  6107 ;% 01   LIT/LITRL,LITRL/0AA,ROT/ZLIT4,ALPCTL/WX_S.Q_0,RDM_WB
                                     0152  6111
                                     0152  6112 BEGIN_CPU_DATA
                                     0002  6113
                                     0002  6114 DCS_DATA
                                     0001  6115
U 00, 7700,C800,0364,0300,0470,1801  0001  6116 ;% 00   NOP
U 01, 6700,5800,DF47,0355,0470,1802  000C  6120 ;% 01   LIT/LITRL,LITRL/0AA,ROT/ZLIT4,ALPCTL/WX_S.Q_0,RDM_WB
U 02, 7300,4800,0364,0300,0470,1803  0017  6124 ;% 02   STOP.CLOCK
                                     0022  6128
                                     0022  6129 END_CPU_DATA
                                     0152  6130
                                     0152  6131 END_TEST_DATA
                                     0152
                                     0152       ;-------------------------------------------------------------------------
                                     0152  6132
                                     0152  6133 ENDTEST
```

```
003C        .SBTTL  "TEST   055     Rotator Test 35 One Extend and Rotate Left Literal Test
003C  6135 ;*******************************************************************************
003C  6136 ;++
003C  6137 ;
003C  6138 ; FUNCTIONAL DESCRIPTION
003C  6139 ;
003C  6140 ;       This test examines the rotator's ability to one extend a nine bit
003C  6141 ;       literal and rotate it left by a specific number of bits.
003C  6142 ;
003C  6143 ; TEST ALGORITHM
003C  6144 ;
003C  6145 ;        1. Initialize the CPU.
003C  6146 ;        2. Load the index K with a 1.
003C  6147 ;        3. Start signature analysis.
003C  6148 ;        4. Loop on I for 3 ROT/OLIT micro-orders.
003C  6149 ;        5. Load the ROT field of the test micro-instruction with the proper
003C  6150 ;           OLIT micro-order as read from a table indexed by I.
003C  6151 ;        6. Loop on J for 5 stuck at zero or one test patterns.
003C  6152 ;        7. Generate a stuck at zero or one test pattern based on J.
003C  6153 ;        8. Load the LITRL field of the test micro-instr. with the pattern.
003C  6154 ;        9. Load the modified test micro-instruction into DCS.
003C  6155 ;       10. Loop on error from here.
003C  6156 ;       11. Execute the micro-code to:
003C  6157 ;               1. Zero extend and left rotate the test pattern in the LITRL
003C  6158 ;                  field by the number of bits specified by the OLIT micro-
003C  6159 ;                  order in the ROT field. Write the result to WHREG.
003C  6160 ;       12. Does the WHREG match the expected data from a table indexed by K?
003C  6161 ;       13. If not, signal an error.
003C  6162 ;       14. Increment the index K for the next test pattern.
003C  6163 ;       15. Continue the loop on J for the remaining test patterns.
003C  6164 ;       16. Continue the loop on I for the remaining ROT field micro-orders.
003C  6165 ;       17. Stop signature analysis.
003C  6166 ;       18. End of this test, skip to the next test.
003C  6167 ;
003C  6168 ; TEST DATA
003C  6169 ;
003C  6170 ;       The expected data for this test consists of the five stuck at zero or
003C  6171 ;       one test patterns one extended and left rotated the proper number of
003C  6172 ;       bits for each of the three OLIT micro-orders tested.
003C  6173 ;
003C  6174 ; LOGIC DESCRIPTION
003C  6175 ;
003C  6176 ;       The logic used to perform the one extension and left rotation of the
003C  6177 ;       LITRL field is internal to the SRM slices. The SRK gate array decodes
003C  6178 ;       the ROT field of the micro-word to generate the control signals,
003C  6179 ;       SHF<4:0>, PRI<1:0>, and SEC<5:0> that control the SRM slices.
003C  6180 ;
003C  6181 ; ASSUMPTIONS
003C  6182 ;
003C  6183 ;       This test assumes the following logic has been tested and is working;
003C  6184 ;       the SBUS and WBUS and the WMUX_S.Q_0 ALPCTL data path.
003C  6185 ;
003C  6186 ; ERROR DESCRIPTION
003C  6187 ;
003C  6188 ;       On error this test will call out the SRM and SRK gate arrays.
```

```
                        003C  6189 ;
                        003C  6190 ;       The following data will also by typed:
                        003C  6191 ;
                        003C  6192 ;       EXPECTED WBUS DATA
                        003C  6193 ;       RECEIVED WBUS DATA
                        003C  6194 ;       THE NUMBER OF THE OLIT MICRO-ORDER UNDER TEST (INDEX I)
                        003C  6195 ;       THE NUMBER OF THE TEST PATTERN BEING USED (INDEX J)
                        003C  6196 ;       THE INDEX INTO THE EXPECTED DATA TABLE (INDEX K)
                        003C  6197 ;
                        003C  6198 ;--
                        003C  6199 ;*******************************************************************************
                        003C  6200 ;///////////////////////////////////////////////////////////////////////////////
                        003C  6201
                        003C  6202         INITIALIZE                                      ; INITIALIZE THE CPU
                        003E  6203         LDINDEX         K,1                             ; LOAD INDEX K WITH A 1
                        0045  6204         LOOP            I,1,3                           ; LOOP ON I FOR 3 OLIT ORDERS
                        004C  6205         LDFIELD         2$,I,B,20$,58,63                ; ROT FIELD GETS AN OLIT ORDER
                        005D  6206         LOOP            J,1,5                           ; LOOP ON J FOR 5 TEST PATTERNS
                        0064  6207         SA01PAT         J,1$                            ; GENERATE A SA01 TEST PATTERN
                        006C  6208         LDFIELD         1$,,W,20$,31,39                 ; LITRL FIELD GETS A PATTERN
                        007D  6209         LOADDCS         20$,,1,1                        ; LOAD TEST INST. INTO DCS
                        0084  6210         ERRLOOP                                         ; LOOP ON ERROR FROM HERE
                        0086  6211         FETCH           0                               ; EXECUTE THE ROT/OLIT AND
                        008B  6212         BRSTCLK         2                               ; WRITE THE RESULT TO WHREG
                        008F  6213         CMPREG          WHREG,10$,K,L                   ; IS WHREG CORRECT?
                        0098  6214         IFERROR         ,<<SRM><SRK>>                   ; IF NOT, SIGNAL AN ERROR
                        009F  6215         INCINDEX        K                               ; INCREMENT K FOR NEXT PATTERN
                        00A2  6216         ENDLOOP         J                               ; CONTINUE J FOR OTHER PATTERNS
                        00A5  6217         ENDLOOP         I                               ; CONTINUE I FOR OTHER ORDERS
                        00A8  6218         SKIP                                            ; END OF TEST SKIP TO NEXT TEST
                        00AF  6219
                        00AF  6220 BEGIN_TEST_DATA
                        00AF
                        00AF       ;-------------------------------------------------------------------------
                        00AF  6221
                00000000 00AF  6222 1$:     .LONG   0       ; LOCATION TO HOLD SA01 PATTERN FOR WDREG
                        00B3  6223
                        00B3  6224 ; ROT field micro-orders
                        00B3  6225
                     3E 00B3  6226 2$:      .BYTE   ^X3E    ; OLIT8         I = 1
                     3C 00B4  6227          .BYTE   ^X3C    ; OLIT16        I = 2
                     3A 00B5  6228          .BYTE   ^X3A    ; OLIT24        I = 3
                        00B6  6229
                        00B6  6230 ; Expected Data
                        00B6  6231
FFFF33FF FFFF55FF FFFEAAFF 00B6  6232 10$:     .LONG   ^XFFFEAAFF,^XFFFF55FF,^XFFFF33FF ; ROT/OLIT8    K = 01-03
         FFFEFFFF FFFF0FFF 00C2  6233          .LONG   ^XFFFF0FFF,^XFFFEFFFF            ;              K = 04-05
FF33FFFF FF55FFFF FEAAFFFF 00CA  6234          .LONG   ^XFEAAFFFF,^XFF55FFFF,^XFF33FFFF ; ROT/OLIT16   K = 06-08
         FEFFFFFF FF0FFFFF 00D6  6235          .LONG   ^XFF0FFFFF,^XFEFFFFFF            ;              K = 09-0A
33FFFFFF 55FFFFFF AAFFFFFE 00DE  6236          .LONG   ^XAAFFFFFE,^X55FFFFFF,^X33FFFFFF ; ROT/OLIT24   K = 0B-0D
         FFFFFFFE 0FFFFFFF 00EA  6237          .LONG   ^X0FFFFFFF,^XFFFFFFFE            ;              K = 0E-0F
                        00F2  6238
                        00F2  6239 ; Test micro-instruction
                        00F2  6240
                        00F2  6241 20$:
```

```
U 01, 6700,5800,FB47,0355,0470,1802  00F2  6242 ;% 01   LIT/LITRL,LITRL/0AA,ROT/OLIT8,ALPCTL/WX_S.Q_0,RDM_WB
                                     00FD  6246
                                     00FD  6247 BEGIN_CPU_DATA
                                     0002  6248
                                     0002  6249 DCS_DATA
                                     0001  6250
U 00, 7700,C800,0364,0300,0470,1801  0001  6251 ;% 00   NOP
U 01, 6700,5800,FB47,0355,0470,1802  000C  6255 ;% 01   LIT/LITRL,LITRL/0AA,ROT/OLIT8,ALPCTL/WX_S.Q_0,RDM_WB
U 02, 7300,4800,0364,0300,0470,1803  0017  6259 ;% 02   STOP.CLOCK
                                     0022  6263
                                     0022  6264 END_CPU_DATA
                                     00FD  6265
                                     00FD  6266 END_TEST_DATA
                                     00FD
                                     00FD       ;-------------------------------------------------------------------------
                                     00FD  6267
                                     00FD  6268 ENDTEST
```

```
0042          .SBTTL  "TEST   056     Rotator Test 36 Arithmetic Right Shift of MBUS by PLATCH Tes
0042  6270 ;*******************************************************************************
0042  6271 ;++
0042  6272 ;
0042  6273 ; FUNCTIONAL DESCRIPTION
0042  6274 ;
0042  6275 ;       This test examines the rotator's ability to perform an arithmetic right
0042  6276 ;       shift of the MBUS by the number of bits specified in the PLATCH.
0042  6277 ;
0042  6278 ; TEST ALGORITHM
0042  6279 ;
0042  6280 ;       This test is implemented in four subtests. Subtest 1 makes sure that no
0042  6281 ;       shifts occur when zero is loaded into the PLATCH and the input data
0042  6282 ;       consists of a longword with only bit 31 set. Subtest 2 uses the arith-
0042  6283 ;       metic right shift function to generate rightward marching one patterns
0042  6284 ;       by using the contents of the loop counter I to load the PLATCH and the
0042  6285 ;       input data used in Subtest 1. Subtest 3 makes sure that no shifts occur
0042  6286 ;       when zero is loaded into the PLATCH and the input data consists of a
0042  6287 ;       longword with only bit 31 clear. Subtest 4 uses the arithmetic right
0042  6288 ;       shift function to generate rightward marching zero patterns by using
0042  6289 ;       the contents of the loop counter I to load the PLATCH and the input
0042  6290 ;       data used in Subtest 3. The test algorithms for Subtests 1 and 3 and
0042  6291 ;       Subtests 2 and 4 are specified below:
0042  6292 ;
0042  6293 ;       Subtests 1 and 3
0042  6294 ;
0042  6295 ;        1. Initialize the CPU. (Subtest 1 only)
0042  6296 ;        2. Start signature analysis. (Subtest 1 only)
0042  6297 ;        3. Load the WDREG with a longword equal to 8000000 (Subtest 1) or
0042  6298 ;           7FFFFFF (Subtest 2).
0042  6299 ;        4. Zero the LITRL field of a PLATCH gets LITRL micro-instruction.
0042  6300 ;        5. Load the micro-instruction into DCS.
0042  6301 ;        6. Loop on error from here.
0042  6302 ;        7. Execute the micro-code to:
0042  6303 ;               1. Load the PLATCH with zero.
0042  6304 ;               2. Load MTEMP0 with the contents of WDREG.
0042  6305 ;               3. Shift MBUS right by PLATCH and write the result to WHREG.
0042  6306 ;        8. Compare the contents of WHREG with the expected data.
0042  6307 ;        9. If the two do not match, signal an error.
0042  6308 ;
0042  6309 ;       Subtests 2 and 4
0042  6310 ;
0042  6311 ;        1. Loop on I for 31 shifts.
0042  6312 ;        2. Save the contents of I.
0042  6313 ;        3. Load the LITRL field of a PLATCH gets LITRL microword with I.
0042  6314 ;        4. Load the microword into DCS.
0042  6315 ;        5. Loop on error from here.
0042  6316 ;        6. Execute the shift and write the result to WHREG.
0042  6317 ;        7. Compare the contents of WHREG with the expected data.
0042  6318 ;        8. If the two do not match, signal an error.
0042  6319 ;        9. Continue the loop on I until all shifts have been tried.
0042  6320 ;       10. Stop signature analysis. (Subtest 4 only)
0042  6321 ;       11. End of this test, skip to next test. (Subtest 4 only)
0042  6322 ;
0042  6323 ; TEST DATA
```

```
                        6324 ;
                        6325 ;       This test uses marching zero and one patterns to test the shifter.
                        6326 ;
                        6327 ; LOGIC DESCRIPTION
                        6328 ;
                        6329 ;       The right shift function is implemented by the ALP and SRM gate arrays
                        6330 ;       which obtain their control signals from ALK and SRK.
                        6331 ;
                        6332 ; ASSUMPTIONS
                        6333 ;
                        6334 ;       This test assumes the following logic has been tested and is working;
                        6335 ;       the scratchpads TEMP0, TEMP1 and TEMP2; the RBUS, MBUS, WBUS and SBUS;
                        6336 ;       the PLATCH and the ALPCTL data path WX_S.Q_0.
                        6337 ;
                        6338 ; ERROR DESCRIPTION
                        6339 ;
                        6340 ;       On error any Subtest will call out the ALP, ALK, SRM and SRK chips.
                        6341 ;
                        6342 ;       The following data will be typed for Subtests 1 and 3:
                        6343 ;
                        6344 ;       EXPECTED WBUS DATA
                        6345 ;       RECEIVED WBUS DATA
                        6346 ;       INPUT DATA (MTEMP0)
                        6347 ;       CONTENTS OF THE PLATCH (MTEMP1)
                        6348 ;
                        6349 ;       This data typed out for Subtests 2 and 4 is:
                        6350 ;
                        6351 ;       EXPECTED WBUS DATA
                        6352 ;       RECEIVED WBUS DATA
                        6353 ;       LOOP COUNTER I
                        6354 ;       INPUT DATA (MTEMP0)
                        6355 ;       CONTENTS OF THE PLATCH (MTEMP1)
                        6356 ;
                        6357 ;--
                        6358 ;*****************************************************************************
                        6359
                        6360         SUBTEST         1
                        6361
                             ;/////////////////////////////////////////////////////////////////////////////
                        6362 ;+
                        6363 ; Checks the right shift function with PLATCH = 0 and MBUS = 80000000.
                        6364 ;-
                        6365         INITIALIZE                                      ; INITIALIZE THE CPU
                        6366         LOADREG         WDREG,3$,,L                     ; LOAD THE WDREG WITH 80000000
                        6367         LDFIELD         2$,,W,10$,31,39                 ; PUT ZERO INTO THE PLATCH
                        6368         LOADDCS         10$,,1,1                        ; LOAD PL_[0] INST. INTO DCS
                        6369         ERRLOOP                                         ; LOOP ON ERROR FROM HERE
                        6370         FETCH           0                               ; EXECUTE THE SHIFT AND WRITE
                        6371         BRSTCLK         5                               ; THE RESULT TO WHREG
                        6372         CMPREG          WHREG,5$,,L                     ; IS THE WHREG CORRECT?
                        6373         IFERROR         2,<<ALP><ALK><SRM><SRK>>; IF NOT, SIGNAL AN ERROR
                        6374
                        6375         SUBTEST         2
                        6376
                             ;/////////////////////////////////////////////////////////////////////////////
```

```
                0087  6375 ;+
                0087  6376 ; Checks the right shifting of MBUS by the PLATCH using marching ones.
                0087  6377 ;-
                0087  6378         LOOP            I,1,31                      ; DO 31 MARCHING ONE PATTERNS
                008E  6379         SAVEINDEX       I,1$                        ; SAVE THE CONTENTS OF I
                0093  6380         LDFIELD         1$,,W,10$,31,39             ; LITRL FIELD GETS I
                00A4  6381         LOADDCS         10$,,1,1                    ; LOAD THE PL_[I] INTO DCS
                00AB  6382         ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                00AD  6383         FETCH           0                           ; EXECUTE THE SHIFT AND WRITE
                00B2  6384         BRSTCLK         5                           ; THE RESULT TO WHREG
                00B6  6385         CMPREG          WHREG,5$,I,L                ; IS THE WHREG CORRECT?
                00BF  6386         IFERROR         2,<<ALP><ALK><SRM><SRK>>    ; IF NOT, SIGNAL AN ERROR
                00C8  6387         ENDLOOP         I                           ; LOOP BACK FOR OTHER PATTERNS
                00CB  6388
                00CB  6389         SUBTEST         3
                00CE
                00CE       ;////////////////////////////////////////////////////////////////////////////
                00CE  6390 ;+
                00CE  6391 ; Checks the right shift function with PLATCH = 0 and MBUS = 7FFFFFF.
                00CE  6392 ;-
                00CE  6393         LOADREG         WDREG,4$,,L                 ; LOAD WDREG WITH 7FFFFFFF
                00D6  6394         LDFIELD         2$,,W,10$,31,39             ; ZERO THE LITRL FIELD
                00E7  6395         LOADDCS         10$,,1,1                    ; LOAD THE PL_[0] INTO DCS
                00EE  6396         ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                00F0  6397         FETCH           0                           ; EXECUTE THE SHIFT AND WRITE
                00F5  6398         BRSTCLK         5                           ; THE RESULT TO WHREG
                00F9  6399         CMPREG          WHREG,6$,,L                 ; IS THE WHREG CORRECT?
                0102  6400         IFERROR         2,<<ALP><ALK><SRM><SRK>>    ; IF NOT, SIGNAL AN ERROR
                010B  6401
                010B  6402         SUBTEST         4
                010E
                010E       ;////////////////////////////////////////////////////////////////////////////
                010E  6403 ;+
                010E  6404 ; Checks the right shifting of MBUS by the PLATCH using marching zeros.
                010E  6405 ;-
                010E  6406         LOOP            I,1,31                      ; DO 31 MARCHING ZEROS
                0115  6407         SAVEINDEX       I,1$                        ; SAVE THE CONTENTS OF I
                011A  6408         LDFIELD         1$,,W,10$,31,39             ; LITRL FIELD GETS I
                012B  6409         LOADDCS         10$,,1,1                    ; LOAD PL_[I] INTO DCS
                0132  6410         ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                0134  6411         FETCH           0                           ; EXECUTE THE SHIFT AND WRITE
                0139  6412         BRSTCLK         5                           ; THE RESULT TO WHREG
                013D  6413         CMPREG          WHREG,6$,I,L                ; IS THE WHREG CORRECT?
                0146  6414         IFERROR         2,<<ALP><ALK><SRM><SRK>>    ; IF NOT, SIGNAL AN ERROR
                014F  6415         ENDLOOP         I                           ; LOOP BACK FOR OTHER PATTERNS
                0152  6416         SKIP                                        ; END OF TEST
                0159  6417
                0159  6418 BEGIN_TEST_DATA
                0159
                0159       ;-------------------------------------------------------------------------
                0159  6419
      0000      0159  6420 1$:     .WORD   0       ; TEMPORARY FOR STORING I INTO
      0000      015B  6421 2$:     .WORD   0       ; CONSTANT 0
                015D  6422
  80000000      015D  6423 3$:     .LONG   ^X80000000      ; DATA FOR MARCHING ONES
```

```
                        7FFFFFFF  0161  6424 4$:     .LONG   ^X7FFFFFFF      ; DATA FOR MARCHING ZEROS
                                  0165  6425
        E0000000 C0000000 80000000  0165  6426 5$:     .LONG   ^X80000000,^XC0000000,^XE0000000        ; I = 01-03
        FC000000 F8000000 F0000000  0171  6427          .LONG   ^XF0000000,^XF8000000,^XFC000000        ; I = 04-06
        FF800000 FF000000 FE000000  017D  6428          .LONG   ^XFE000000,^XFF000000,^XFF800000        ; I = 07-09
        FFF00000 FFE00000 FFC00000  0189  6429          .LONG   ^XFFC00000,^XFFE00000,^XFFF00000        ; I = 0A-0C
        FFFE0000 FFFC0000 FFF80000  0195  6430          .LONG   ^XFFF80000,^XFFFC0000,^XFFFE0000        ; I = 0D-0F
        FFFFC000 FFFF8000 FFFF0000  01A1  6431          .LONG   ^XFFFF0000,^XFFFF8000,^XFFFFC000        ; I = 10-12
        FFFFF800 FFFFF000 FFFFE000  01AD  6432          .LONG   ^XFFFFE000,^XFFFFF000,^XFFFFF800        ; I = 13-15
        FFFFFF00 FFFFFE00 FFFFFC00  01B9  6433          .LONG   ^XFFFFFC00,^XFFFFFE00,^XFFFFFF00        ; I = 16-18
        FFFFFFE0 FFFFFFC0 FFFFFF80  01C5  6434          .LONG   ^XFFFFFF80,^XFFFFFFC0,^XFFFFFFE0        ; I = 19-1B
        FFFFFFFC FFFFFFF8 FFFFFFF0  01D1  6435          .LONG   ^XFFFFFFF0,^XFFFFFFF8,^XFFFFFFFC        ; I = 1C-1E
                 FFFFFFFF FFFFFFFE  01DD  6436          .LONG   ^XFFFFFFFE,^XFFFFFFFF                   ; I = 1F-20
                                  01E5  6437
        1FFFFFFF 3FFFFFFF 7FFFFFFF  01E5  6438 6$:     .LONG   ^X7FFFFFFF,^X3FFFFFFF,^X1FFFFFFF        ; I = 01-03
        03FFFFFF 07FFFFFF 0FFFFFFF  01F1  6439          .LONG   ^X0FFFFFFF,^X07FFFFFF,^X03FFFFFF        ; I = 04-06
        007FFFFF 00FFFFFF 01FFFFFF  01FD  6440          .LONG   ^X01FFFFFF,^X00FFFFFF,^X007FFFFF        ; I = 07-09
        000FFFFF 001FFFFF 003FFFFF  0209  6441          .LONG   ^X003FFFFF,^X001FFFFF,^X000FFFFF        ; I = 0A-0C
        0001FFFF 0003FFFF 0007FFFF  0215  6442          .LONG   ^X0007FFFF,^X0003FFFF,^X0001FFFF        ; I = 0D-0F
        00003FFF 00007FFF 0000FFFF  0221  6443          .LONG   ^X0000FFFF,^X00007FFF,^X00003FFF        ; I = 10-12
        000007FF 00000FFF 00001FFF  022D  6444          .LONG   ^X00001FFF,^X00000FFF,^X000007FF        ; I = 13-15
        000000FF 000001FF 000003FF  0239  6445          .LONG   ^X000003FF,^X000001FF,^X000000FF        ; I = 16-18
        0000001F 0000003F 0000007F  0245  6446          .LONG   ^X0000007F,^X0000003F,^X0000001F        ; I = 19-1B
        00000003 00000007 0000000F  0251  6447          .LONG   ^X0000000F,^X00000007,^X00000003        ; I = 1C-1E
                 00000000 00000001  025D  6448          .LONG   ^X00000001,^X00000000                   ; I = 1F-20
                                  0265  6449
                                  0265  6450 10$:
U 01, 7700,1800,EF64,0300,0470,1802  0265  6451 ;% 01   PL_[0]  ; LOAD THE PLATCH WITH ZERO OR I
                                  0270  6455
                                  0270  6456 BEGIN_CPU_DATA
                                  0002  6457
                                  0002  6458 DCS_DATA
                                  0001  6459
U 00, 7700,C800,0364,0300,0470,1801  0001  6460 ;% 00   NOP
U 01, 7700,1800,EF64,0300,0470,1802  000C  6464 ;% 01   PL_[0]
U 02, 5700,4860,0364,0300,0470,1803  0017  6468 ;% 02   M[TEMP0]_RDM
U 03, 6700,C800,0F47,0308,0470,1804  0022  6472 ;% 03   MSRC/TEMP0,RSRC/TEMP2,ROT/ASR.M.P,ALPCTL/WX_S.Q_0,RDM_WB
U 04, 7700,4861,B370,0300,0470,1805  002D  6476 ;% 04   M[TEMP1]_PL
U 05, 7300,4800,0364,0300,0470,1806  0038  6480 ;% 05   STOP.CLOCK
                                  0043  6484
                                  0043  6485 RTEMP_DATA
                                  0001  6486
                         00000000 0001  6487          .LONG   0       ; INPUT DATA TO ROTATE
                         00000000 0005  6488          .LONG   0       ; CONTENTS OF THE PLATCH
                         00000000 0009  6489          .LONG   0       ; CONSTANT ZERO TO SOURCE ONTO RBUS
                                  000D  6490
                                  000D  6491 END_CPU_DATA
                                  0270  6492
                                  0270  6493 END_TEST_DATA
                                  0270
                                  0270        ;-------------------------------------------------------------------------
                                  0270  6494
                                  0270  6495 ENDTEST
```

```
0043        .SBTTL  "TEST   057     Rotator Test 37 Arithmetic Right Shift of MBUS by -PLATCH Te
0043  6497 ;*******************************************************************************
0043  6498 ;++
0043  6499 ;
0043  6500 ; FUNCTIONAL DESCRIPTION
0043  6501 ;
0043  6502 ;       This test examines the arithmetic right shift of MBUS data by the
0043  6503 ;       number of bits specified by the twos complement of the PLATCH.
0043  6504 ;
0043  6505 ; TEST ALGORITHM
0043  6506 ;
0043  6507 ;       This test is implemented in two subtests. The first subtest floats a
0043  6508 ;       one through the PLATCH, by looping six times on the index I.
0043  6509 ;       The algorithm for both subtests is as follows:
0043  6510 ;
0043  6511 ;        1. Initialize the CPU. (Subtest 1 only)
0043  6512 ;        2. Start signature analysis.
0043  6513 ;        3. Loop on I for 6 floating one (Subtest 1) or floating zero
0043  6514 ;           (Subtest 2) patterns.
0043  6515 ;        4. Load a floating one or zero into the LITRL field of a PL_[] inst.
0043  6516 ;        5. Load the instruction into DCS.
0043  6517 ;        6. Loop on error from here.
0043  6518 ;        7. Execute the micro-code to:
0043  6519 ;                1. Load the PLATCH with the floating pattern.
0043  6520 ;                2. Execute the rotation and write the result to WHREG.
0043  6521 ;                3. Read the PLATCH into MTEMP1.
0043  6522 ;        8. Compare the WHREG with the expected WBUS results.
0043  6523 ;        9. If the two do not match, signal an error.
0043  6524 ;       10. Continue the loop on I for the remaining test patterns.
0043  6525 ;       11. End signature analysis.
0043  6526 ;       12. End of the test. (Subtest 2 only)
0043  6527 ;
0043  6528 ; TEST DATA
0043  6529 ;
0043  6530 ;       The following data is used to implement this test:
0043  6531 ;
0043  6532 ;            Index I       PLATCH value        WBUS Data
0043  6533 ;
0043  6534 ;               01              00              80000000
0043  6535 ;               02              01              FFFFFFFF
0043  6536 ;               03              02              FFFFFFFE
0043  6537 ;               04              04              FFFFFFF8
0043  6538 ;               05              08              FFFFFF80
0043  6539 ;               06              10              FFFF8000
0043  6540 ;
0043  6541 ; LOGIC DESCRIPTION
0043  6542 ;
0043  6543 ;       This test examines the two's complementer inside the SRK gate array as
0043  6544 ;       well as the shifter inside the ALP slices.
0043  6545 ;
0043  6546 ; ASSUMPTIONS
0043  6547 ;
0043  6548 ;       This test assumes the following logic has been tested and is working;
0043  6549 ;       the scratchpads TEMP0, TEMP1 and TEMP2; the RBUS, MBUS, SBUS and WBUS;
0043  6550 ;       the PLATCH; and the ALPCTL WX_S.Q_0 data path.
```

```
                        0043  6551 ;
                        0043  6552 ; ERROR DESCRIPTION
                        0043  6553 ;
                        0043  6554 ;       On error both subtests will call out the ALP, ALK, SRK and SRM chips.
                        0043  6555 ;
                        0043  6556 ;       The following data will also by typed:
                        0043  6557 ;
                        0043  6558 ;       EXPECTED WBUS DATA
                        0043  6559 ;       RECEIVED WBUS DATA
                        0043  6560 ;       LOOP COUNTER I
                        0043  6561 ;       INPUT DATA TO BE SHIFTED (MTEMP0)
                        0043  6562 ;       CONTENTS OF THE PLATCH (MTEMP1)
                        0043  6563 ;
                        0043  6564 ;--
                        0043  6565 ;*******************************************************************************
                        0043  6566
                        0043  6567         SUBTEST         1
                        0046
                        0046       ;////////////////////////////////////////////////////////////////////////////
                        0046  6568 ;+
                        0046  6569 ; Examine the ROT/ASR.M.-P micro-order by floating a one through the PLATCH.
                        0046  6570 ;-
                        0046  6571         INITIALIZE                                      ; INITIALIZE THE CPU
                        0048  6572         LOOP            I,1,6                           ; LOOP ON I FOR 6 FLOATING ONES
                        004F  6573         LDFIELD         1$,I,B,10$,31,39                ; LITRL FIELD GETS A FLOATING 1
                        0060  6574         LOADDCS         10$,,1,1                        ; LOAD PLATCH GETS LITRL TO DCS
                        0067  6575         ERRLOOP                                         ; LOOP ON ERROR FROM HERE
                        0069  6576         FETCH           0                               ; EXECUTE THE SHIFT AND WRITE
                        006E  6577         BRSTCLK         4                               ; THE RESULT TO WHREG
                        0072  6578         CMPREG          WHREG,3$,I,L                    ; IS WHREG CORRECT?
                        007B  6579         IFERROR         2,<<ALP><ALK><SRM><SRK>>        ; IF NOT, SIGNAL AN ERROR
                        0084  6580         ENDLOOP         I                               ; LOOP BACK FOR OTHER FLOATS
                        0087  6581
                        0087  6582         SUBTEST         2
                        008A
                        008A       ;////////////////////////////////////////////////////////////////////////////
                        008A  6583 ;+
                        008A  6584 ; Examine the ROT/ASR.M.-P micro-order by floating a zero through the PLATCH.
                        008A  6585 ;-
                        008A  6586         LOOP            I,1,6                           ; LOOP ON I FOR 6 FLOATING 0'S
                        0091  6587         LDFIELD         2$,I,B,10$,31,39                ; LITRL FIELD GETS A FLOATING 0
                        00A2  6588         LOADDCS         10$,,1,1                        ; LOAD PLATCH GETS LITRL TO DCS
                        00A9  6589         ERRLOOP                                         ; LOOP ON ERROR FROM HERE
                        00AB  6590         FETCH           0                               ; EXECUTE THE SHIFT AND WRITE
                        00B0  6591         BRSTCLK         4                               ; THE RESULT TO WHREG
                        00B4  6592         CMPREG          WHREG,4$,I,L                    ; IS WHREG CORRECT?
                        00BD  6593         IFERROR         2,<<ALP><ALK><SRM><SRK>>        ; IF NOT, SIGNAL AN ERROR
                        00C6  6594         ENDLOOP         I                               ; LOOP BACK FOR OTHER FLOATS
                        00C9  6595         SKIP                                            ; END OF TEST
                        00D0  6596
                        00D0  6597 BEGIN_TEST_DATA
                        00D0
                        00D0       ;-------------------------------------------------------------------------
                        00D0  6598
10 08 04 02 01 00       00D0  6599 1$:      .BYTE   ^X00,^X01,^X02,^X04,^X08,^X10   ; FLOATING 1 FOR PLATCH
```

```
                                00D6  6600
                0F 17 1B 1D 1E 1F  00D6  6601 2$:     .BYTE   ^X1F,^X1E,^X1D,^X1B,^X17,^X0F    ; FLOATING 0 FOR PLATCH
                                00DC  6602
        FFFFFFFE FFFFFFFF 80000000  00DC  6603 3$:     .LONG   ^X80000000,^XFFFFFFFF,^XFFFFFFFE ; EXPECTED DATA FOR FLOATING 1
        FFFF8000 FFFFFF80 FFFFFFF8  00E8  6604         .LONG   ^XFFFFFFF8,^XFFFFFF80,^XFFFF8000
                                00F4  6605
        F0000000 E0000000 C0000000  00F4  6606 4$:     .LONG   ^XC0000000,^XE0000000,^XF0000000 ; EXPECTED DATA FOR FLOATING 0
        FFFFC000 FFC00000 FC000000  0100  6607         .LONG   ^XFC000000,^XFFC00000,^XFFFFC000
                                010C  6608
                                010C  6609 10$:
U 01, 7700,1800,EF64,0300,0470,1802  010C  6610 ;% 01   PL_[0] ; MICRO-WORD TO LOAD THE PLATCH WITH A FLOATING PATTERN
                                0117  6614
                                0117  6615 BEGIN_CPU_DATA
                                0002  6616
                                0002  6617 DCS_DATA
                                0001  6618
U 00, 7700,C800,0364,0300,0470,1801  0001  6619 ;% 00   NOP
U 01, 7700,1800,EF64,0300,0470,1802  000C  6623 ;% 01   PL_[0]
U 02, 6700,0800,AB47,0308,0470,1803  0017  6627 ;% 02   MSRC/TEMP0,RSRC/TEMP2,ROT/ASR.M.-P,ALPCTL/WX_S.Q_0,RDM_WB
U 03, 7700,C861,B370,0300,0470,1804  0022  6631 ;% 03   M[TEMP1]_PL
U 04, 7300,4800,0364,0300,0470,1805  002D  6635 ;% 04   STOP.CLOCK
                                0038  6639
                                0038  6640 RTEMP_DATA
                                0001  6641
                        80000000  0001  6642         .LONG   ^X80000000      ; MTEMP0 = VALUE FOR MBUS
                        00000000  0005  6643         .LONG   0               ; MTEMP1 = CONTENTS OF THE PLATCH
                        00000000  0009  6644         .LONG   0               ; MTEMP2 = CONSTANT ZERO FOR RBUS
                                000D  6645
                                000D  6646 END_CPU_DATA
                                0117  6647
                                0117  6648 END_TEST_DATA
                                0117
                                0117       ;-----------------------------------------------------------------------
                                0117  6649
                                0117  6650 ENDTEST
```

```
                        003D        .SBTTL  "TEST   058     Rotator Test 38 Arithmetic Right Shift of MBUS by 3 Test
                        003D  6652 ;*******************************************************************************
                        003D  6653 ;++
                        003D  6654 ;
                        003D  6655 ; FUNCTIONAL DESCRIPTION
                        003D  6656 ;
                        003D  6657 ;       This test examines the ROT/ASR.M.3 micro-order to make sure that the
                        003D  6658 ;       MBUS is shifted right by three bits.
                        003D  6659 ;
                        003D  6660 ; TEST ALGORITHM
                        003D  6661 ;
                        003D  6662 ;       This test is implemented by the following algorithm:
                        003D  6663 ;
                        003D  6664 ;        1. Initialize the CPU.
                        003D  6665 ;        2. Start signature analysis.
                        003D  6666 ;        3. Loop on error from here.
                        003D  6667 ;        4. Execute the micro-code to:
                        003D  6668 ;               1. Shift the contents of MTEMP0 right by three bits and write
                        003D  6669 ;                  the result into the WHREG.
                        003D  6670 ;        5. Compare the contents of the WHREG against the expected WBUS data.
                        003D  6671 ;        6. If the two do not match, signal an error.
                        003D  6672 ;        7. Stop signature analysis.
                        003D  6673 ;        8. End of test, skip to the next test.
                        003D  6674 ;
                        003D  6675 ; TEST DATA
                        003D  6676 ;
                        003D  6677 ;       The data used for this test consists of a longword equal to 80000000
                        003D  6678 ;       hex which is right shifted by 3 bits to make F0000000 hex.
                        003D  6679 ;
                        003D  6680 ; LOGIC DESCRIPTION
                        003D  6681 ;
                        003D  6682 ;       The logic used to decode a ROT/ASM.M.3 micro-order resides inside the
                        003D  6683 ;       SRK gate array.
                        003D  6684 ;
                        003D  6685 ; ASSUMPTIONS
                        003D  6686 ;
                        003D  6687 ;       This test assumes the following logic has been tested and is working;
                        003D  6688 ;       the MTEMP0 scratchpad; the MBUS, SBUS and WBUS; and the ALPCTL data
                        003D  6689 ;       path WX_S.Q_0.
                        003D  6690 ;
                        003D  6691 ; ERROR DESCRIPTION
                        003D  6692 ;
                        003D  6693 ;       On error this test will call out the SRK gate array and type out the
                        003D  6694 ;       following information:
                        003D  6695 ;
                        003D  6696 ;       EXPECTED WBUS DATA
                        003D  6697 ;       RECEIVED WBUS DATA
                        003D  6698 ;
                        003D  6699 ;--
                        003D  6700 ;*******************************************************************************
                        003D  6701 ;///////////////////////////////////////////////////////////////////////////////
                        003D  6702
                        003D  6703         INITIALIZE                                  ; INITIALIZE THE CPU
                        003F  6704         ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                        0041  6705         FETCH           0                           ; EXECUTE THE SHIFT AND WRITE
```

```
                        0046  6706          BRSTCLK         2                       ; THE RESULT TO WHREG
                        004A  6707          CMPREG          WHREG,1$,,L             ; IS WHREG CORRECT?
                        0053  6708          IFERROR         ,<<SRK>>                ; IF NOT, SIGNAL ERROR
                        0059  6709          SKIP                                    ; END OF TEST
                        0060  6710
                        0060  6711 BEGIN_TEST_DATA
                        0060
                        0060          ;-------------------------------------------------------------------------
                        0060  6712
                F0000000  0060  6713 1$:      .LONG    ^XF0000000      ; EXPECTED RESULT OF RIGHT SHIFT
                        0064  6714
                        0064  6715 BEGIN_CPU_DATA
                        0002  6716
                        0002  6717 DCS_DATA
                        0001  6718
U 00, 7700,C800,0364,0300,0470,1801  0001  6719 ;% 00    NOP
U 01, 6700,8800,7747,0300,0470,1802  000C  6723 ;% 01    MSRC/TEMP0,ROT/ASR.M.3,ALPCTL/WX_S.Q_0,RDM_WB
U 02, 7300,4800,0364,0300,0470,1803  0017  6727 ;% 02    STOP.CLOCK
                        0022  6731
                        0022  6732 RTEMP_DATA
                        0001  6733
                80000000  0001  6734          .LONG    ^X80000000      ; INPUT DATA FOR THE SHIFT
                        0005  6735
                        0005  6736 END_CPU_DATA
                        0064  6737
                        0064  6738 END_TEST_DATA
                        0064
                        0064          ;-------------------------------------------------------------------------
                        0064  6739
                        0064  6740 ENDTEST
```

```
                        0041        .SBTTL  "TEST   059     Rotator Test 39 Arithmetic Left Shift of MBUS by PLATCH Test
                        0041   6742 ;*******************************************************************************
                        0041   6743 ;++
                        0041   6744 ;
                        0041   6745 ; FUNCTIONAL DESCRIPTION
                        0041   6746 ;
                        0041   6747 ;       This test insures that the MBUS can be left shifted by the PLATCH.
                        0041   6748 ;
                        0041   6749 ; TEST ALGORITHM
                        0041   6750 ;
                        0041   6751 ;       This test is implemented by floating a one from MBUS<0> to the WBUS
                        0041   6752 ;       <31:00> by the following algorithm:
                        0041   6753 ;
                        0041   6754 ;        1. Initialize the CPU.
                        0041   6755 ;        2. Start signature analysis.
                        0041   6756 ;        3. Loop on I for 32 floating one patterns.
                        0041   6757 ;        4. Save the contents of I.
                        0041   6758 ;        5. Place the contents of I into the LITRL field of a PL_[] instr.
                        0041   6759 ;        6. Write the PL_[I] instruction into DCS.
                        0041   6760 ;        7. Loop on error from here.
                        0041   6761 ;        8. Execute the micro-code to:
                        0041   6762 ;                1. Load the PLATCH with I.
                        0041   6763 ;                2. Rotate the MBUS left by the PLATCH (ROT/ASL.M.P) and write
                        0041   6764 ;                   the result to WHREG.
                        0041   6765 ;        9. Compare the contents of WHREG with the expected data from a table
                        0041   6766 ;           indexed by I.
                        0041   6767 ;       10. If the two do not match, signal an error.
                        0041   6768 ;       11. Continue the loop on I for the remaining floating ones.
                        0041   6769 ;       12. Stop signature analysis.
                        0041   6770 ;       13. End of this test, skip to the next test.
                        0041   6771 ;
                        0041   6772 ; TEST DATA
                        0041   6773 ;
                        0041   6774 ;       This test uses floating one patterns for the ROT/ASL.M.P micro-order.
                        0041   6775 ;
                        0041   6776 ; LOGIC DESCRIPTION
                        0041   6777 ;
                        0041   6778 ;       The data to be shifted is passed through the SRM bit slices to the ALP
                        0041   6779 ;       chips where the shifting is performed. The SRK and ALK gate arrays
                        0041   6780 ;       generate the signals necessary to control this operation.
                        0041   6781 ;
                        0041   6782 ; ASSUMPTIONS
                        0041   6783 ;
                        0041   6784 ;       This test assumes the following logic has been tested and is working;
                        0041   6785 ;       The scratchpad MTEMP0; the MBUS, SBUS and WBUS; the PLATCH; and the
                        0041   6786 ;       ALPCTL data path WX_S.Q_0.
                        0041   6787 ;
                        0041   6788 ; ERROR DESCRIPTION
                        0041   6789 ;
                        0041   6790 ;       On error this test will call out the ALP, ALK, SRM and SRK gate arrays.
                        0041   6791 ;       The following data will also be typed:
                        0041   6792 ;
                        0041   6793 ;       EXPECTED WBUS DATA
                        0041   6794 ;       RECEIVED WBUS DATA
                        0041   6795 ;       LOOP COUNT I
```

```
                                0041  6796 ;       INPUT DATA PATTERN (MTEMP0)
                                0041  6797 ;       CONTENTS OF THE PLATCH (MTEMP1)
                                0041  6798 ;--
                                0041  6799 ;*******************************************************************************
                                0041  6800 ;///////////////////////////////////////////////////////////////////////////////
                                0041  6801
                                0041  6802         INITIALIZE                                      ; INITIALIZE THE CPU
                                0043  6803         LOOP            I,1,32                          ; LOOP ON I FOR 32 FLOATING 1'S
                                004A  6804         SAVEINDEX       I,1$                            ; SAVE THE CONTENTS OF I
                                004F  6805         LDFIELD         1$,,W,10$,31,39                 ; LITRL FIELD OF PL_[] GETS I
                                0060  6806         LOADDCS         10$,,1,1                        ; LOAD PL_[I] INTO DCS
                                0067  6807         ERRLOOP                                         ; LOOP ON ERROR FROM HERE
                                0069  6808         FETCH           0                               ; EXECUTE THE ROT/ASL.M.P AND
                                006E  6809         BRSTCLK         4                               ; WRITE THE RESULT TO WHREG
                                0072  6810         CMPREG          WHREG,2$,I,L                    ; IS WHREG CORRECT?
                                007B  6811         IFERROR         2,<<ALP><ALK><SRM><SRK>>        ; IF NOT, SIGNAL AN ERROR
                                0084  6812         ENDLOOP         I                               ; CONTINUE I FOR REST OF FLOATS
                                0087  6813         SKIP                                            ; END OF TEST
                                008E  6814
                                008E  6815 BEGIN_TEST_DATA
                                008E
                                008E       ;-------------------------------------------------------------------
                                008E  6816
                          0000  008E  6817 1$:      .WORD   0       ; TEMPORARY FOR STORING I INTO
                                0090  6818
00000004 00000002 00000000      0090  6819 2$:      .LONG   ^X00000000,^X00000002,^X00000004 ; EXPECTED DATA TABLE
00000020 00000010 00000008      009C  6820          .LONG   ^X00000008,^X00000010,^X00000020
00000100 00000080 00000040      00A8  6821          .LONG   ^X00000040,^X00000080,^X00000100
00000800 00000400 00000200      00B4  6822          .LONG   ^X00000200,^X00000400,^X00000800
00004000 00002000 00001000      00C0  6823          .LONG   ^X00001000,^X00002000,^X00004000
00020000 00010000 00008000      00CC  6824          .LONG   ^X00008000,^X00010000,^X00020000
00100000 00080000 00040000      00D8  6825          .LONG   ^X00040000,^X00080000,^X00100000
00800000 00400000 00200000      00E4  6826          .LONG   ^X00200000,^X00400000,^X00800000
04000000 02000000 01000000      00F0  6827          .LONG   ^X01000000,^X02000000,^X04000000
20000000 10000000 08000000      00FC  6828          .LONG   ^X08000000,^X10000000,^X20000000
         80000000 40000000      0108  6829          .LONG   ^X40000000,^X80000000
                                0110  6830
                                0110  6831 10$:
U 01, 7700,1800,EF64,0300,0470,1802 0110  6832 ;% 01    PL_[0]  ; MICRO-INSTRUCTION TO LOAD THE PLATCH WITH I
                                011B  6836
                                011B  6837 BEGIN_CPU_DATA
                                0002  6838
                                0002  6839 DCS_DATA
                                0001  6840
U 00, 7700,C800,0364,0300,0470,1801 0001  6841 ;% 00    NOP
U 01, 7700,1800,EF64,0300,0470,1802 000C  6845 ;% 01    PL_[0]
U 02, 6700,4800,A747,0300,0470,1803 0017  6849 ;% 02    MSRC/TEMP0,ROT/ASL.M.P,ALPCTL/WX_S.Q_0,RDM_WB
U 03, 7700,C861,B370,0300,0470,1804 0022  6853 ;% 03    M[TEMP1]_PL
U 04, 7300,4800,0364,0300,0470,1805 002D  6857 ;% 04    STOP.CLOCK
                                0038  6861
                                0038  6862 RTEMP_DATA
                                0001  6863
                    00000001    0001  6864          .LONG   ^X00000001      ; MTEMP0 = PATTERN TO BE SHIFTED
                    00000000    0005  6865          .LONG   0               ; MTEMP1 = CONTENTS OF THE PLATCH
                                0009  6866
```

```
                0009  6867 END_CPU_DATA
                011B  6868
                011B  6869 END_TEST_DATA
                011B
                011B       ;-----------------------------------------------------------------------------
                011B  6870
                011B  6871 ENDTEST
```

```
                        0041        .SBTTL  "TEST   05A     Rotator Test 40 Arithmetic Left Shift of RBUS by PLATCH Test
                        0041  6873 ;*******************************************************************************
                        0041  6874 ;++
                        0041  6875 ;
                        0041  6876 ; FUNCTIONAL DESCRIPTION
                        0041  6877 ;
                        0041  6878 ;       This test insures that the RBUS can be left shifted by the PLATCH.
                        0041  6879 ;
                        0041  6880 ; TEST ALGORITHM
                        0041  6881 ;
                        0041  6882 ;       This test is implemented by floating a one from RBUS<0> through to
                        0041  6883 ;       WBUS<31:00> by using the following algorithm:
                        0041  6884 ;
                        0041  6885 ;        1. Initialize the CPU.
                        0041  6886 ;        2. Start signature analysis.
                        0041  6887 ;        3. Loop on I for 32 floating one patterns.
                        0041  6888 ;        4. Save the contents of I.
                        0041  6889 ;        5. Place the contents of I into the LITRL field of a PL_[] instr.
                        0041  6890 ;        6. Write the PL_[I] instruction into DCS.
                        0041  6891 ;        7. Loop on error from here.
                        0041  6892 ;        8. Execute the micro-code to:
                        0041  6893 ;               1. Load the PLATCH with I.
                        0041  6894 ;               2. Rotate the RBUS left by the PLATCH (ROT/ASL.R.P) and write
                        0041  6895 ;                  the result to WHREG.
                        0041  6896 ;        9. Compare the contents of WHREG with the expected data from a table
                        0041  6897 ;           indexed by I.
                        0041  6898 ;       10. If the two do not match, signal an error.
                        0041  6899 ;       11. Continue the loop on I for the remaining floating ones.
                        0041  6900 ;       12. Stop signature analysis.
                        0041  6901 ;       13. End of this test, skip to the next test.
                        0041  6902 ;
                        0041  6903 ; TEST DATA
                        0041  6904 ;
                        0041  6905 ;       This test uses floating one patterns for the ROT/ASL.R.P micro-order.
                        0041  6906 ;
                        0041  6907 ; LOGIC DESCRIPTION
                        0041  6908 ;
                        0041  6909 ;       The data to be shifted is passed through the SRM bit slices to the ALP
                        0041  6910 ;       chips where the shifting is performed. The SRK and ALK gate arrays
                        0041  6911 ;       generate the signals necessary to control this operation.
                        0041  6912 ;
                        0041  6913 ; ASSUMPTIONS
                        0041  6914 ;
                        0041  6915 ;       This test assumes the following logic has been tested and is working;
                        0041  6916 ;       The scratchpad RTEMP0; the RBUS, SBUS and WBUS; the PLATCH; and the
                        0041  6917 ;       ALPCTL data path WX_S.Q_0.
                        0041  6918 ;
                        0041  6919 ; ERROR DESCRIPTION
                        0041  6920 ;
                        0041  6921 ;       On error this test will call out the ALP, ALK, SRM and SRK gate arrays.
                        0041  6922 ;       The following data will also be typed:
                        0041  6923 ;
                        0041  6924 ;       EXPECTED WBUS DATA
                        0041  6925 ;       RECEIVED WBUS DATA
                        0041  6926 ;       LOOP COUNT I
```

```
                                0041  6927 ;       INPUT DATA PATTERN (RTEMP0)
                                0041  6928 ;       CONTENTS OF THE PLATCH (MTEMP1)
                                0041  6929 ;--
                                0041  6930 ;*******************************************************************************
                                0041  6931 ;///////////////////////////////////////////////////////////////////////////////
                                0041  6932
                                0041  6933         INITIALIZE                                      ; INITIALIZE THE CPU
                                0043  6934         LOOP            I,1,32                          ; LOOP ON I FOR 32 FLOATING 1'S
                                004A  6935         SAVEINDEX       I,1$                            ; SAVE THE CONTENTS OF I
                                004F  6936         LDFIELD         1$,,W,10$,31,39                 ; LITRL FIELD OF PL_[] GETS I
                                0060  6937         LOADDCS         10$,,1,1                        ; LOAD PL_[I] INTO DCS
                                0067  6938         ERRLOOP                                         ; LOOP ON ERROR FROM HERE
                                0069  6939         FETCH           0                               ; EXECUTE THE ROT/ASL.R.P AND
                                006E  6940         BRSTCLK         4                               ; WRITE THE RESULT TO WHREG
                                0072  6941         CMPREG          WHREG,2$,I,L                    ; IS WHREG CORRECT?
                                007B  6942         IFERROR         2,<<ALP><ALK><SRM><SRK>>        ; IF NOT, SIGNAL AN ERROR
                                0084  6943         ENDLOOP         I                               ; CONTINUE I FOR REST OF FLOATS
                                0087  6944         SKIP                                            ; END OF TEST
                                008E  6945
                                008E  6946 BEGIN_TEST_DATA
                                008E
                                008E       ;-------------------------------------------------------------------------
                                008E  6947
                          0000  008E  6948 1$:     .WORD   0       ; TEMPORARY FOR STORING I INTO
                                0090  6949
 00000004 00000002 00000000     0090  6950 2$:     .LONG   ^X00000000,^X00000002,^X00000004        ; EXPECTED DATA
 00000020 00000010 00000008     009C  6951         .LONG   ^X00000008,^X00000010,^X00000020
 00000100 00000080 00000040     00A8  6952         .LONG   ^X00000040,^X00000080,^X00000100
 00000800 00000400 00000200     00B4  6953         .LONG   ^X00000200,^X00000400,^X00000800
 00004000 00002000 00001000     00C0  6954         .LONG   ^X00001000,^X00002000,^X00004000
 00020000 00010000 00008000     00CC  6955         .LONG   ^X00008000,^X00010000,^X00020000
 00100000 00080000 00040000     00D8  6956         .LONG   ^X00040000,^X00080000,^X00100000
 00800000 00400000 00200000     00E4  6957         .LONG   ^X00200000,^X00400000,^X00800000
 04000000 02000000 01000000     00F0  6958         .LONG   ^X01000000,^X02000000,^X04000000
 20000000 10000000 08000000     00FC  6959         .LONG   ^X08000000,^X10000000,^X20000000
          80000000 40000000     0108  6960         .LONG   ^X40000000,^X80000000
                                0110  6961
                                0110  6962 10$:
U 01, 7700,1800,EF64,0300,0470,1802  0110  6963 ;% 01   PL_[0]  ; MICRO-INSTRUCTION TO LOAD THE PLATCH WITH I
                                011B  6967
                                011B  6968 BEGIN_CPU_DATA
                                0002  6969
                                0002  6970 DCS_DATA
                                0001  6971
U 00, 7700,C800,0364,0300,0470,1801  0001  6972 ;% 00   NOP
U 01, 7700,1800,EF64,0300,0470,1802  000C  6976 ;% 01   PL_[0]
U 02, 6700,0800,A347,0300,0470,1803  0017  6980 ;% 02   RSRC/TEMP0,ROT/ASL.R.P,ALPCTL/WX_S.Q_0,RDM_WB
U 03, 7700,C861,B370,0300,0470,1804  0022  6984 ;% 03   M[TEMP1]_PL
U 04, 7300,4800,0364,0300,0470,1805  002D  6988 ;% 04   STOP.CLOCK
                                0038  6992
                                0038  6993 RTEMP_DATA
                                0001  6994
                    00000001    0001  6995         .LONG   ^X00000001      ; RTEMP0 = PATTERN TO BE SHIFTED
                    00000000    0005  6996         .LONG   0               ; MTEMP1 = CONTENTS OF THE PLATCH
                                0009  6997
```

```
                        0009  6998 END_CPU_DATA
                        011B  6999
                        011B  7000 END_TEST_DATA
                        011B
                        011B       ;-----------------------------------------------------------------------------
                        011B  7001
                        011B  7002 ENDTEST
```

```
                        003F        .SBTTL  "TEST   05B     Rotator Test 41 Arithmetic Left Shift of RBUS by Size Test
                        003F  7004 ;*******************************************************************************
                        003F  7005 ;++
                        003F  7006 ;
                        003F  7007 ; FUNCTIONAL DESCRIPTION
                        003F  7008 ;
                        003F  7009 ;       This test examines the arithmetic left shift of the RBUS by the number
                        003F  7010 ;       of bits specified by the DTYPE field of the micro-instruction.
                        003F  7011 ;
                        003F  7012 ; TEST ALGORITHM
                        003F  7013 ;
                        003F  7014 ;       This test is implemented by shifting six stuck at zero or one test
                        003F  7015 ;       patterns by the three data types available from the DTYPE field.
                        003F  7016 ;       The algorithm is as follows:
                        003F  7017 ;
                        003F  7018 ;        1. Initialize the CPU.
                        003F  7019 ;        2. Start signature analysis.
                        003F  7020 ;        3. Initialize the index K with a 1.
                        003F  7021 ;        4. Loop on I for 3 data types. (Byte, Word and Long)
                        003F  7022 ;        5. Load the DTYPE field of the test micro-instruction with a data
                        003F  7023 ;           type from a table indexed by I.
                        003F  7024 ;        6. Load the test micro-instruction into DCS.
                        003F  7025 ;        7. Loop on J for 6 stuck at zero or one test patterns.
                        003F  7026 ;        8. Generate a stuck at zero or one pattern based on J.
                        003F  7027 ;        9. Load the test pattern into the WDREG.
                        003F  7028 ;       10. Loop on error from here.
                        003F  7029 ;       11. Execute the micro-code to:
                        003F  7030 ;               1. Load RTEMP0 with the WDREG.
                        003F  7031 ;               2. Shift RTEMP0 left by the number of bits specified by DTYPE
                        003F  7032 ;                  (using ROT/ASL.R.SIZ), strobe the VBUS to latch in DSIZE and
                        003F  7033 ;                  write the result to WHREG.
                        003F  7034 ;       12. Compare the WHREG with the expected data from a table indexed by K.
                        003F  7035 ;       13. If the two do not match, signal an error.
                        003F  7036 ;       14. Increment the index K for the next test pattern.
                        003F  7037 ;       15. Continue the loop on J for the remaining stuck at patterns.
                        003F  7038 ;       16. Continue the loop on I for the remaining DTYPES.
                        003F  7039 ;       17. Stop signature analysis.
                        003F  7040 ;       18. End of test, skip to the next test.
                        003F  7041 ;
                        003F  7042 ; TEST DATA
                        003F  7043 ;
                        003F  7044 ;       Three bytes are used to load the DTYPE field. Their values are 0, 1 and
                        003F  7045 ;       2 which correspond to the data types Byte, Word and Long.
                        003F  7046 ;       The table below gives the I,J,K, DTYPE, RBUS and WBUS values:
                        003F  7047 ;
                        003F  7048 ;       I J K   DTYPE     RBUS            WBUS
                        003F  7049 ;
                        003F  7050 ;       1 1 1   BYTE    AAAAAAAA        AAAAAAAA
                        003F  7051 ;       1 2 2   BYTE    55555555        55555555
                        003F  7052 ;       1 3 3   BYTE    33333333        33333333
                        003F  7053 ;       1 4 4   BYTE    0F0F0F0F        0F0F0F0F
                        003F  7054 ;       1 5 5   BYTE    00FF00FF        00FF00FF
                        003F  7055 ;       1 6 6   BYTE    0000FFFF        0000FFFF
                        003F  7056 ;       2 1 7   WORD    AAAAAAAA        55555554
                        003F  7057 ;       2 2 8   WORD    55555555        AAAAAAAA
```

```
003F  7058 ;       2 3 9   WORD    33333333        66666666
003F  7059 ;       2 4 A   WORD    0F0F0F0F        1E1E1E1E
003F  7060 ;       2 5 B   WORD    00FF00FF        01FE01FE
003F  7061 ;       2 6 C   WORD    0000FFFF        0001FFFE
003F  7062 ;       3 1 D   LONG    AAAAAAAA        AAAAAAA8
003F  7063 ;       3 2 E   LONG    55555555        55555554
003F  7064 ;       3 3 F   LONG    33333333        CCCCCCCC
003F  7065 ;       3 4 10  LONG    0F0F0F0F        3C3C3C3C
003F  7066 ;       3 5 11  LONG    00FF00FF        03FC03FC
003F  7067 ;       3 6 12  LONG    0000FFFF        0003FFFC
003F  7068 ;
003F  7069 ; LOGIC DESCRIPTION
003F  7070 ;
003F  7071 ;       The logic involved with this test is inside the SRK gate array.
003F  7072 ;
003F  7073 ; ASSUMPTIONS
003F  7074 ;
003F  7075 ;       This test assumes the following logic has been tested and is working;
003F  7076 ;       The scratchpad RTEMP0; the RBUS, SBUS and WBUS; the discrete logic used
003F  7077 ;       to decode the DTYPE field into the DSIZE control signals; and the
003F  7078 ;       ALPCTL data patch WX_S.Q_0.
003F  7079 ;
003F  7080 ; ERROR DESCRIPTION
003F  7081 ;
003F  7082 ;       On error this test will call out the SRK gate array and will type out
003F  7083 ;       one the following data:
003F  7084 ;
003F  7085 ;       EXPECTED WBUS DATA
003F  7086 ;       RECEIVED WBUS DATA
003F  7087 ;       DTYPE LOOP COUNT (INDEX I)
003F  7088 ;       TEST PATTERN LOOP COUNT (INDEX J)
003F  7089 ;       EXPECTED DATA TABLE INDEX (INDEX K)
003F  7090 ;       INPUT SHIFT DATA (RTEMP0)
003F  7091 ;--
003F  7092 ;*******************************************************************************
003F  7093 ;///////////////////////////////////////////////////////////////////////////////
003F  7094
003F  7095         INITIALIZE                                      ; INITIALIZE THE CPU
0041  7096         LDINDEX         K,1                             ; INITIALIZE INDEX K WITH 1
0048  7097         LOOP            I,1,3                           ; LOOP ON I FOR 3 DTYPES
004F  7098         LDFIELD         2$,I,B,10$,40,41                ; LOAD DTYPE FIELD
0060  7099         LOADDCS         10$,,2,1                        ; LOAD TEST INSTR. INTO DCS
0067  7100         LOOP            J,1,6                           ; LOOP ON J FOR 6 SA01 PATTERNS
006E  7101         SA01PAT         J,1$                            ; GENERATE A SA01 TEST PATTERN
0076  7102         LOADREG         WDREG,1$,,L                     ; LOAD TEST PATTERN INTO WDREG
007E  7103         ERRLOOP                                         ; LOOP ON ERROR FROM HERE
0080  7104         FETCH           0                               ; EXECUTE THE ROT/ASL.R.SIZ AND
0085  7105         BRSTCLK         3                               ; WRITE THE RESULT TO WHREG
0089  7106         CMPREG          WHREG,3$,K,L                    ; IS WHREG CORRECT AND
0092  7107         IFERROR         1,<<SRK>>                       ; IN NOT, SIGNAL AN ERROR
0098  7108         INCINDEX        K                               ; INCREMENT K FOR NEXT PATTERN
009B  7109         ENDLOOP         J                               ; CONTINUE J FOR ALL SA01
009E  7110         ENDLOOP         I                               ; CONTINUE I FOR ALL DTYPES
00A1  7111         SKIP                                            ; END OF TEST
00A8  7112
```

```
                                    00A8  7113 BEGIN_TEST_DATA
                                    00A8
                                    00A8        ;-------------------------------------------------------------------------
                                    00A8  7114
                           00000000 00A8  7115 1$:      .LONG   0       ; TEMPORARY FOR SA01PAT
                                    00AC  7116
                                    00AC  7117 ; DSIZE DATA
                                    00AC  7118
                                 00 00AC  7119 2$:      .BYTE   0       ; DTYPE = BYTE
                                 01 00AD  7120          .BYTE   1       ; DTYPE = WORD
                                 02 00AE  7121          .BYTE   2       ; DTYPE = LONGWORD
                                    00AF  7122
                                    00AF  7123 ; EXPECTED DATA
                                    00AF  7124
        33333333 55555555 AAAAAAAA  00AF  7125 3$:      .LONG   ^XAAAAAAAA,^X55555555,^X33333333
        0000FFFF 00FF00FF 0F0F0F0F  00BB  7126          .LONG   ^X0F0F0F0F,^X00FF00FF,^X0000FFFF
        66666666 AAAAAAAA 55555554  00C7  7127          .LONG   ^X55555554,^XAAAAAAAA,^X66666666
        0001FFFE 01FE01FE 1E1E1E1E  00D3  7128          .LONG   ^X1E1E1E1E,^X01FE01FE,^X0001FFFE
        CCCCCCCC 55555554 AAAAAAA8  00DF  7129          .LONG   ^XAAAAAAA8,^X55555554,^XCCCCCCCC
        0003FFFC 03FC03FC 3C3C3C3C  00EB  7130          .LONG   ^X3C3C3C3C,^X03FC03FC,^X0003FFFC
                                    00F7  7131
                                    00F7  7132 ; MICROWORD TO TEST SIZE ROTATION
                                    00F7  7133
                                    00F7  7134 10$:
U 02, 6701,0800,5F47,0000,0470,1803 00F7  7135 ;% 02   RSRC/TEMP0,DTYPE/BYTE,ROT/ASL.R.SIZ,ALPCTL/WX_S.Q_0,RDM_WB,VSIZE/1
                                    0102  7139
                                    0102  7140 BEGIN_CPU_DATA
                                    0002  7141
                                    0002  7142 DCS_DATA
                                    0001  7143
U 00, 7700,C800,0364,0300,0470,1801 0001  7144 ;% 00   NOP
U 01, 5700,8840,0364,0300,0470,1802 000C  7148 ;% 01   R[TEMP0]_RDM
U 02, 6701,0800,5F47,0000,0470,1803 0017  7152 ;% 02   RSRC/TEMP0,DTYPE/BYTE,ROT/ASL.R.SIZ,ALPCTL/WX_S.Q_0,RDM_WB,VSIZE/1
U 03, 7300,C800,0364,0300,0470,1804 0022  7156 ;% 03   STOP.CLOCK
                                    002D  7160
                                    002D  7161 RTEMP_DATA
                                    0001  7162
                           00000000 0001  7163          .LONG   0               ; RTEMP0 = INPUT DATA TO BE SHIFTED
                                    0005  7164
                                    0005  7165 END_CPU_DATA
                                    0102  7166
                                    0102  7167 END_TEST_DATA
                                    0102
                                    0102        ;-------------------------------------------------------------------------
                                    0102  7168
                                    0102  7169 ENDTEST
```

```
003C                    .SBTTL  "TEST   05C     Rotator Test 42 Arithmetic Left Shift of RBUS by 7 Test
003C  7171 ;*****************************************************************************
003C  7172 ;++
003C  7173 ;
003C  7174 ; FUNCTIONAL DESCRIPTION
003C  7175 ;
003C  7176 ;       This test examines an arithmetic left shift of the RBUS by 7 bits.
003C  7177 ;
003C  7178 ; TEST ALGORITHM
003C  7179 ;
003C  7180 ;       This test is implemented by the following algorithm:
003C  7181 ;
003C  7182 ;        1. Initialize the CPU.
003C  7183 ;        2. Start signature analysis.
003C  7184 ;        3. Loop on I for 6 stuck at zero or one test patterns.
003C  7185 ;        4. Generate a stuck at zero or one test pattern based on I.
003C  7186 ;        5. Load the test pattern into the WDREG.
003C  7187 ;        6. Loop on error from here.
003C  7188 ;        7. Execute the micro-code to:
003C  7189 ;               1. Load RTEMP0 with the WDREG.
003C  7190 ;               2. Shift (ROT/ASL.R.7) and write the result to WHREG.
003C  7191 ;        8. Compare WHREG with the expected data from a table indexed by I.
003C  7192 ;        9. If the two do not match, signal an error.
003C  7193 ;       10. Continue the loop on I for the remaining stuck at patterns.
003C  7194 ;       11. Stop signature analysis.
003C  7195 ;       12. End of test, skip to the next test.
003C  7196 ;
003C  7197 ; TEST DATA
003C  7198 ;
003C  7199 ;       This test uses 6 stuck at zero or one patterns shifted left by 7 bits.
003C  7200 ;
003C  7201 ; LOGIC DESCRIPTION
003C  7202 ;
003C  7203 ;       The arithmetic left shift of RBUS by 7 bits is decoded inside the SRK
003C  7204 ;       gate array.
003C  7205 ;
003C  7206 ; ASSUMPTIONS
003C  7207 ;
003C  7208 ;       This test assumes the following logic has been tested and is working;
003C  7209 ;       the scratchpad RTEMP0; the RBUS, SBUS and WBUS; and the ALPCTL data
003C  7210 ;       path WX_S.Q_0.
003C  7211 ;
003C  7212 ; ERROR DESCRIPTION
003C  7213 ;
003C  7214 ;       On error this test will call out the SRK gate array and will type out
003C  7215 ;       the following data:
003C  7216 ;
003C  7217 ;       EXPECTED WBUS DATA
003C  7218 ;       RECEIVED WBUS DATA
003C  7219 ;       TEST PATTERN LOOP COUNT (INDEX I)
003C  7220 ;       INPUT SHIFT DATA (RTEMP0)
003C  7221 ;--
003C  7222 ;*****************************************************************************
003C  7223 ;/////////////////////////////////////////////////////////////////////////////
003C  7224
```

```
                                  003C          INITIALIZE                                  ; INITIALIZE THE CPU
                                  003E          LOOP            I,1,6                       ; LOOP ON I FOR 6 SA01 PATTERNS
                                  0045          SA01PAT         I,1$                        ; GENERATE A SA01 TEST PATTERN
                                  004D          LOADREG         WDREG,1$,,L                 ; LOAD TEST PATTERN INTO WDREG
                                  0055          ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                                  0057          FETCH           0                           ; EXECUTE THE ROT/ASL.R.7 AND
                                  005C          BRSTCLK         3                           ; WRITE THE RESULT TO WHREG
                                  0060          CMPREG          WHREG,2$,I,L                ; IS WHREG CORRECT?
                                  0069          IFERROR         1,<<SRK>>                   ; IF NOT, SIGNAL AN ERROR
                                  006F          ENDLOOP         I                           ; CONTINUE I FOR ALL PATTERNS
                                  0072          SKIP                                        ; END OF TEST
                                  0079
                                  0079  BEGIN_TEST_DATA
                                  0079
                                  0079      ;-----------------------------------------------------------------------
                                  0079
                        00000000  0079  1$:     .LONG   0       ; LOCATION TO HOLD SA01PAT FOR WDREG
                                  007D
                                  007D  ; Expected Data
                                  007D
        99999980 AAAAAA80 55555500  007D  2$:     .LONG   ^X55555500,^XAAAAAA80,^X99999980
        007FFF80 7F807F80 87878780  0089          .LONG   ^X87878780,^X7F807F80,^X007FFF80
                                  0095
                                  0095  BEGIN_CPU_DATA
                                  0002
                                  0002  DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 5700,8840,0364,0300,0470,1802  000C  ;% 01   R[TEMP0]_RDM
U 02, 6700,4800,5747,0300,0470,1803  0017  ;% 02   RSRC/TEMP0,ROT/ASL.R.7,ALPCTL/WX_S.Q_0,RDM_WB
U 03, 7300,C800,0364,0300,0470,1804  0022  ;% 03   STOP.CLOCK
                                  002D
                                  002D  RTEMP_DATA
                                  0001
                        00000000  0001          .LONG   0       ; RTEMP0 = INPUT DATA TO SHIFT LEFT
                                  0005
                                  0005  END_CPU_DATA
                                  0095
                                  0095  END_TEST_DATA
                                  0095
                                  0095      ;-----------------------------------------------------------------------
                                  0095
                                  0095  ENDTEST
```

```
0038          .SBTTL  "TEST   05D     Rotator Test 43 Unpack Floating Point Fraction Test
0038  7277 ;*******************************************************************************
0038  7278 ;++
0038  7279 ;
0038  7280 ; FUNCTIONAL DESCRIPTION
0038  7281 ;
0038  7282 ;       This test examines the rotator's ability to unpack the fraction field
0038  7283 ;       of a floating point datum. The output data is generated by performing
0038  7284 ;       an extract and merge operation on data input from the MBUS concatenated
0038  7285 ;       with the RBUS.
0038  7286 ;
0038  7287 ; TEST ALGORITHM
0038  7288 ;
0038  7289 ;       This test is implemented by running seven stuck at zero or one test
0038  7290 ;       patterns through the 64 bits of the MBUS concatenated with the RBUS.
0038  7291 ;       The test algorithm executes as follows:
0038  7292 ;
0038  7293 ;        1. Initialize the CPU.
0038  7294 ;        2. Start signature analysis.
0038  7295 ;        3. Load the micro-code for the first six test patterns.
0038  7296 ;        4. Loop on I for 6 stuck at zero or one test patterns.
0038  7297 ;        5. Generate a stuck a zero or one test pattern for a longword.
0038  7298 ;        6. Load the pattern into the WDREG.
0038  7299 ;        7. Loop on error from here.
0038  7300 ;        8. Execute the micro-code to:
0038  7301 ;                1. Load the test pattern from WDREG to MTEMP0.
0038  7302 ;                2. Load the test pattern from WDREG to RTEMP1.
0038  7303 ;                3. Placing RTEMP1 on the RBUS and MTEMP0 on the MBUS, execute
0038  7304 ;                   the unpack function and route the result on the SBUS through
0038  7305 ;                   the ALP's WMUX to the RDM's WHREG.
0038  7306 ;        9. Compare the contents of the WHREG against a table of expected data.
0038  7307 ;       10. If the two do not match, signal an error.
0038  7308 ;       11. Continue the loop started at step 4 for the remaining patterns.
0038  7309 ;       12. Load the micro-code for the seventh test pattern.
0038  7310 ;       13. Loop once on I = 7 for the last stuck at zero or one test pattern.
0038  7311 ;       14. Loop on error from here.
0038  7312 ;       15. Execute the micro-code to:
0038  7313 ;                4. Load the LONLIT register with all ones.
0038  7314 ;                5. Load MTEMP0 with zero.
0038  7315 ;                6. Load MTEMP1 (and RTEMP1) with the LONLIT register.
0038  7316 ;                7. Placing RTEMP1 on the RBUS and MTEMP0 on the MBUS, execute
0038  7317 ;                   the unpack function and route the result on the SBUS through
0038  7318 ;                   the ALP's WMUX to the RDM's WHREG.
0038  7319 ;       16. Compare the contents of the WHREG against a table of expected data.
0038  7320 ;       17. If the two do not match, signal an error.
0038  7321 ;       18. End the loop started at step 13.
0038  7322 ;       19. Stop signature analysis.
0038  7323 ;       20. End of this test, skip to the next test.
0038  7324 ;
0038  7325 ; TEST DATA
0038  7326 ;
0038  7327 ;       Seven stuck at zero or one test patterns are used to test the unpack
0038  7328 ;       function. The unpack extracts and merges fields from the MBUS con-
0038  7329 ;       catenated with the RBUS and places the result on the SBUS as follows:
0038  7330 ;
```

```
0038  7331 ;                SBUS <06:00> <-- RBUS <31:25>
0038  7332 ;                SBUS <22:07> <-- MBUS <31:16>
0038  7333 ;                SBUS <29:23> <-- MBUS <06:00>
0038  7334 ;                SBUS <31:30> <-- 01
0038  7335 ;
0038  7336 ;       The input and output data patterns used are:
0038  7337 ;
0038  7338 ;                     Input Data      Expected Output Data
0038  7339 ;
0038  7340 ;                  MBUS     RBUS          SBUS
0038  7341 ;                 AAAAAAAA AAAAAAAA       55555555
0038  7342 ;                 55555555 55555555       6AAAAAAA
0038  7343 ;                 33333333 33333333       59999999
0038  7344 ;                 0F0F0F0F 0F0F0F0F       47878787
0038  7345 ;                 00FF00FF 00FF00FF       7F807F80
0038  7346 ;                 0000FFFF 0000FFFF       7F800000
0038  7347 ;                 00000000 FFFFFFFF       4000007F
0038  7348 ;
0038  7349 ; LOGIC DESCRIPTION
0038  7350 ;
0038  7351 ;       The unpack function is performed inside the SRM gate arrays. The SRM's
0038  7352 ;       are controlled by the SRK gate array via the control signals PRI<1:0>
0038  7353 ;       and SEC<5:0>.
0038  7354 ;
0038  7355 ; ASSUMPTIONS
0038  7356 ;
0038  7357 ;       This test assumes the following logic on the DPM has been tested and
0038  7358 ;       is working; the MBUS, RBUS, SBUS and WBUS; the scratchpads RTEMP1 and
0038  7359 ;       MTEMP0; and the ALPCTL data path for WX_S.Q_0.
0038  7360 ;
0038  7361 ; ERROR DESCRIPTION
0038  7362 ;
0038  7363 ;       On error this test will call out the SRM and SRK gate arrays. The
0038  7364 ;       following data will also by typed:
0038  7365 ;
0038  7366 ;       EXPECTED SBUS DATA
0038  7367 ;       RECEIVED SBUS DATA
0038  7368 ;       LOOP COUNT I
0038  7369 ;       MBUS DATA
0038  7370 ;       RBUS DATA
0038  7371 ;--
0038  7372 ;*******************************************************************************
0038  7373 ;///////////////////////////////////////////////////////////////////////////////
0038  7374
0038  7375          INITIALIZE                                  ; INITIALIZE THE CPU
003A  7376          LOADDCS         10$,,0,5                    ; LOAD FIRST TEST MICRO-CODE
0041  7377          LOOP            I,1,6                       ; LOOP ON I FOR 6 SA01 PATTERNS
0048  7378          SA01PAT         I,1$                        ; GENERATE A SA01 PATTERN
0050  7379          LOADREG         WDREG,1$,,L                 ; LOAD SA01 PATTERN INTO WDREG
0058  7380          ERRLOOP                                     ; LOOP ON ERROR FROM HERE
005A  7381          FETCH           0                           ; EXECUTE THE UNPACK OPERATION
005F  7382          BRSTCLK         4                           ; AND WRITE THE RESULT TO WHREG
0063  7383          CMPREG          WHREG,2$,I,L                ; WAS THE UNPACK SUCCESSFUL?
006C  7384          IFERROR         2,<<SRM><SRK>>              ; IF NOT, SIGNAL AN ERROR
0073  7385          ENDLOOP         I                           ; CONTINUE LOOP FOR 6 PATTERNS
```

```
                        0076  7386          LOADDCS         20$.,0,6                ; LOAD SECOND TEST MICRO-CODE
                        007D  7387          LOOP            I,7,7                   ; LOOP ON I FOR 7TH PATTERN
                        0084  7388          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        0086  7389          FETCH           0                       ; EXECUTE THE UNPACK
                        008B  7390          BRSTCLK         5                       ; AND WRITE THE RESULT TO WHREG
                        008F  7391          CMPREG          WHREG,2$,I,L            ; WAS THE UNPACK SUCCESSFUL?
                        0098  7392          IFERROR         2,<<SRM><SRK>>          ; IF NOT, SIGNAL AN ERROR
                        009F  7393          ENDLOOP         I                       ; END LOOP FOR 7TH PATTERN
                        00A2  7394          SKIP                                    ; END OF TEST
                        00A9  7395
                        00A9  7396 BEGIN_TEST_DATA
                        00A9
                        00A9        ;-------------------------------------------------------------------------
                        00A9  7397
00000000                00A9  7398 1$:      .LONG   0       ; LOCATION TO HOLD SA01PAT
                        00AD  7399
                        00AD  7400 ;
                        00AD  7401 ; Expected Data
                        00AD  7402 ;
                        00AD  7403
55555555                00AD  7404 2$:      .LONG   ^X55555555
6AAAAAAA                00B1  7405          .LONG   ^X6AAAAAAA
59999999                00B5  7406          .LONG   ^X59999999
47878787                00B9  7407          .LONG   ^X47878787
7F807F80                00BD  7408          .LONG   ^X7F807F80
7F800000                00C1  7409          .LONG   ^X7F800000
4000007F                00C5  7410          .LONG   ^X4000007F
                        00C9  7411
                        00C9  7412 ;
                        00C9  7413 ; Test micro-code
                        00C9  7414 ;
                        00C9  7415
                        00C9  7416 10$:
U 00, 7700,C800,0364,0300,0470,1801   00C9  7417 ;% 00   NOP
U 01, 5700,C860,0364,0300,0470,1802   00D4  7421 ;% 01   M[TEMP0]_RDM
U 02, 5700,4840,0364,0304,0470,1803   00DF  7425 ;% 02   R[TEMP1]_RDM
U 03, 6700,8800,6747,0304,0470,1804   00EA  7429 ;% 03   RSRC/TEMP1,MSRC/TEMP0,ROT/GETFPF,ALPCTL/WX_S.Q_0,RDM_WB
U 04, 7300,4800,0364,0300,0470,1805   00F5  7433 ;% 04   STOP.CLOCK
                        0100  7437
                        0100  7438 20$:
U 00, 7700,C800,0364,0300,0470,1801   0100  7439 ;% 00   NOP
U 01, 7700,B800,0000,0000,0470,1802   010B  7443 ;% 01   LONLIT_[0FFFFFFFF]
U 02, 7700,4860,5BE4,03D8,0470,1803   0116  7447 ;% 02   M[TEMP0]_R[ZERO]
U 03, 7700,8861,5BE4,03D4,0470,1804   0121  7451 ;% 03   M[TEMP1]_R[LONLIT]
U 04, 6700,0800,6747,0304,0470,1805   012C  7455 ;% 04   RSRC/TEMP1,MSRC/TEMP0,ROT/GETFPF,ALPCTL/WX_S.Q_0,RDM_WB
U 05, 7300,4800,0364,0300,0470,1806   0137  7459 ;% 05   STOP.CLOCK
                        0142  7463
                        0142  7464 END_TEST_DATA
                        0142
                        0142        ;-------------------------------------------------------------------------
                        0142  7465
                        0142  7466 ENDTEST
```

```
0034          .SBTTL  "TEST   05E     Rotator Test 44 Pack Floating Point Number Test
0034  7468 ;*******************************************************************************
0034  7469 ;++
0034  7470 ;
0034  7471 ; FUNCTIONAL DESCRIPTION
0034  7472 ;
0034  7473 ;       This test examines the rotator logic used to pack the exponent and
0034  7474 ;       fraction field of a floating point datum. The output data is generated
0034  7475 ;       by performing an extract and merge operation on data input from the
0034  7476 ;       MBUS concatenated with the RBUS.
0034  7477 ;
0034  7478 ; TEST ALGORITHM
0034  7479 ;
0034  7480 ;       This test is implemented by running seven stuck at zero or one test
0034  7481 ;       patterns through the 64 bits of the MBUS concatenated with the RBUS.
0034  7482 ;       The test algorithm executes as follows:
0034  7483 ;
0034  7484 ;        1. Initialize the CPU.
0034  7485 ;        2. Start signature analysis.
0034  7486 ;        3. Load the micro-code for the first six test patterns.
0034  7487 ;        4. Loop on I for 6 stuck at zero or one test patterns.
0034  7488 ;        5. Generate a stuck a zero or one test pattern for a longword.
0034  7489 ;        6. Load the pattern into the WDREG.
0034  7490 ;        7. Loop on error from here.
0034  7491 ;        8. Execute the micro-code to:
0034  7492 ;                1. Load the test pattern from WDREG to MTEMP0.
0034  7493 ;                2. Load the test pattern from WDREG to RTEMP1.
0034  7494 ;                3. Placing RTEMP1 on the RBUS and MTEMP0 on the MBUS, execute
0034  7495 ;                   the pack function and route the result on the SBUS through
0034  7496 ;                   the ALP's WMUX to the RDM's WHREG.
0034  7497 ;        9. Compare the contents of the WHREG against a table of expected data.
0034  7498 ;       10. If the two do not match, signal an error.
0034  7499 ;       11. Continue the loop started at step 4 for the remaining patterns.
0034  7500 ;       12. Load the micro-code for the seventh test pattern.
0034  7501 ;       13. Loop once on I = 7 for the last stuck at zero or one test pattern.
0034  7502 ;       14. Loop on error from here.
0034  7503 ;       15. Execute the micro-code to:
0034  7504 ;                4. Load the LONLIT register with all ones.
0034  7505 ;                5. Load MTEMP0 with zero.
0034  7506 ;                6. Load MTEMP1 (and RTEMP1) with the LONLIT register.
0034  7507 ;                7. Placing RTEMP1 on the RBUS and MTEMP0 on the MBUS, execute
0034  7508 ;                   the pack function and route the result on the SBUS through
0034  7509 ;                   the ALP's WMUX to the RDM's WHREG.
0034  7510 ;       16. Compare the contents of the WHREG against a table of expected data.
0034  7511 ;       17. If the two do not match, signal an error.
0034  7512 ;       18. End the loop started at step 13.
0034  7513 ;       19. Stop signature analysis.
0034  7514 ;       20. End of this test, skip to the next test.
0034  7515 ;
0034  7516 ; TEST DATA
0034  7517 ;
0034  7518 ;       Seven stuck at zero or one test patterns are used to test the pack
0034  7519 ;       function. The pack extracts and merges fields from the MBUS con-
0034  7520 ;       catenated with the RBUS and places the result on the SBUS as follows:
0034  7521 ;
```

```
0034  7522 ;                SBUS <06:00> <-- MBUS <31:25>
0034  7523 ;                SBUS <14:07> <-- RBUS <07:00>
0034  7524 ;                SBUS <31:16> <-- MBUS <24:09>
0034  7525 ;                SBUS <15>    <-- 0
0034  7526 ;
0034  7527 ;       The input and output data patterns used are:
0034  7528 ;
0034  7529 ;                     Input Data      Expected Output Data
0034  7530 ;
0034  7531 ;                   MBUS     RBUS          SBUS
0034  7532 ;                  AAAAAAAA AAAAAAAA       55555555
0034  7533 ;                  55555555 55555555       AAAA2AAA
0034  7534 ;                  33333333 33333333       99991999
0034  7535 ;                  0F0F0F0F 0F0F0F0F       87870787
0034  7536 ;                  00FF00FF 00FF00FF       7F807F80
0034  7537 ;                  0000FFFF 0000FFFF       007F7F80
0034  7538 ;                  00000000 FFFFFFFF       00007F80
0034  7539 ;
0034  7540 ; LOGIC DESCRIPTION
0034  7541 ;
0034  7542 ;       The pack function is performed inside the SRM gate arrays. The SRM's
0034  7543 ;       are controlled by the SRK gate array via the control signals PRI<1:0>
0034  7544 ;       and SEC<5:0>.
0034  7545 ;
0034  7546 ; ASSUMPTIONS
0034  7547 ;
0034  7548 ;       This test assumes the following logic on the DPM has been tested and
0034  7549 ;       is working; the MBUS, RBUS, SBUS and WBUS; the scratchpads RTEMP1 and
0034  7550 ;       MTEMP0; and the ALPCTL data path for WX_S.Q_0.
0034  7551 ;
0034  7552 ; ERROR DESCRIPTION
0034  7553 ;
0034  7554 ;       On error this test will call out the SRM and SRK gate arrays. The
0034  7555 ;       following data will also by typed:
0034  7556 ;
0034  7557 ;       EXPECTED SBUS DATA
0034  7558 ;       RECEIVED SBUS DATA
0034  7559 ;       LOOP COUNT I
0034  7560 ;       MBUS DATA
0034  7561 ;       RBUS DATA
0034  7562 ;--
0034  7563 ;*******************************************************************************
0034  7564 ;///////////////////////////////////////////////////////////////////////////////
0034  7565
0034  7566          INITIALIZE                                  ; INTIALIZE THE CPU
0036  7567          LOADDCS         10$,,0,5                    ; LOAD FIRST MICRO-CODE SET
003D  7568          LOOP            I,1,6                       ; LOOP ON I FOR 6 SA01 PATTERNS
0044  7569          SA01PAT         I,1$                        ; GENERATE A SA01 PATTERN
004C  7570          LOADREG         WDREG,1$,,L                 ; LOAD WDREG WITH THE PATTERN
0054  7571          ERRLOOP                                     ; LOOP ON ERROR FROM HERE
0056  7572          FETCH           0                           ; EXECUTE THE PACK AND WRITE
005B  7573          BRSTCLK         4                           ; THE RESULT TO THE WHREG
005F  7574          CMPREG          WHREG,2$,I,L                ; WAS THE PACK CORRECT?
0068  7575          IFERROR         2,<<SRM><SRK>>              ; IF NOT, SIGNAL AN ERROR
006F  7576          ENDLOOP         I                           ; CONTINUE LOOP FOR 6 PATTERNS
```

```
                        0072  7577          LOADDCS         20$.,0,6                ; LOAD SECOND MICRO-CODE SET
                        0079  7578          LOOP            I,7,7                   ; LOOP ON I FOR 7TH PATTERN
                        0080  7579          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        0082  7580          FETCH           0                       ; EXECUTE THE PACK AND WRITE
                        0087  7581          BRSTCLK         5                       ; THE RESULT TO THE WHREG
                        008B  7582          CMPREG          WHREG,2$,I,L            ; WAS THE PACK CORRECT?
                        0094  7583          IFERROR         2,<<SRM><SRK>>          ; IF NOT, SIGNAL AN ERROR
                        009B  7584          ENDLOOP         I                       ; END LOOP FOR 7TH PATTERN
                        009E  7585          SKIP                                    ; END OF TEST
                        00A5  7586
                        00A5  7587 BEGIN_TEST_DATA
                        00A5
                        00A5          ;-------------------------------------------------------------------------
                        00A5  7588
00000000                00A5  7589 1$:      .LONG   0       ; LOCATION TO HOLD SA01PAT
                        00A9  7590
                        00A9  7591 ;
                        00A9  7592 ; Expected Data
                        00A9  7593 ;
                        00A9  7594
55555555                00A9  7595 2$:      .LONG   ^X55555555
AAAA2AAA                00AD  7596          .LONG   ^XAAAA2AAA
99991999                00B1  7597          .LONG   ^X99991999
87870787                00B5  7598          .LONG   ^X87870787
7F807F80                00B9  7599          .LONG   ^X7F807F80
007F7F80                00BD  7600          .LONG   ^X007F7F80
00007F80                00C1  7601          .LONG   ^X00007F80
                        00C5  7602
                        00C5  7603 ;
                        00C5  7604 ; First test micro-code set
                        00C5  7605 ;
                        00C5  7606
                        00C5  7607 10$:
U 00, 7700,C800,0364,0300,0470,1801   00C5  7608 ;% 00   NOP
U 01, 5700,C860,0364,0300,0470,1802   00D0  7612 ;% 01   M[TEMP0]_RDM
U 02, 5700,4840,0364,0304,0470,1803   00DB  7616 ;% 02   R[TEMP1]_RDM
U 03, 6700,8800,6B47,0304,0470,1804   00E6  7620 ;% 03   RSRC/TEMP1,MSRC/TEMP0,ROT/FPACK,ALPCTL/WX_S.Q_0,RDM_WB
U 04, 7300,4800,0364,0300,0470,1805   00F1  7624 ;% 04   STOP.CLOCK
                        00FC  7628
                        00FC  7629 20$:
U 00, 7700,C800,0364,0300,0470,1801   00FC  7630 ;% 00   NOP
U 01, 7700,B800,0000,0000,0470,1802   0107  7634 ;% 01   LONLIT_[0FFFFFFFF]
U 02, 7700,4860,5BE4,03D8,0470,1803   0112  7638 ;% 02   M[TEMP0]_R[ZERO]
U 03, 7700,8861,5BE4,03D4,0470,1804   011D  7642 ;% 03   M[TEMP1]_R[LONLIT]
U 04, 6700,0800,6B47,0304,0470,1805   0128  7646 ;% 04   RSRC/TEMP1,MSRC/TEMP0,ROT/FPACK,ALPCTL/WX_S.Q_0,RDM_WB
U 05, 7300,4800,0364,0300,0470,1806   0133  7650 ;% 05   STOP.CLOCK
                        013E  7654
                        013E  7655 END_TEST_DATA
                        013E
                        013E          ;-------------------------------------------------------------------------
                        013E  7656
                        013E  7657 ENDTEST
```

```
0030          .SBTTL  "TEST   05F     Rotator Test 45 Floating Point Literal Test
0030  7659 ;*******************************************************************************
0030  7660 ;++
0030  7661 ;
0030  7662 ; FUNCTIONAL DESCRIPTION
0030  7663 ;
0030  7664 ;       This test will examine the rotator's ability to exapnd a floating
0030  7665 ;       point literal on the MBUS.
0030  7666 ;
0030  7667 ; TEST ALGORITHM
0030  7668 ;
0030  7669 ;       This test is implemented using six longword stuck at zero or one test
0030  7670 ;       patterns. The patterns are written onto the MBUS and expanded through
0030  7671 ;       the rotator using the FPLIT function to the SBUS. The contents of the
0030  7672 ;       SBUS are then routed through the ALP's to the WBUS via the ALPCTL's
0030  7673 ;       WX_S.Q_0 micro-order where it can be sampled by the RDM's WHREG.
0030  7674 ;       The test executes as follows:
0030  7675 ;
0030  7676 ;        1. Initialize the CPU.
0030  7677 ;        2. Start signature analysis.
0030  7678 ;        3. Loop on I for 6 longword stuck at zero or one test patterns.
0030  7679 ;        4. Generate a longword stuck at zero or one test pattern based on I.
0030  7680 ;        5. Load the pattern into the WDREG.
0030  7681 ;        6. Loop on error from here.
0030  7682 ;        7. Execute the micro-code to:
0030  7683 ;                1. Load the WDREG into MTEMP0.
0030  7684 ;                2. Source MTEMP0 onto the MBUS and zero onto the RBUS and
0030  7685 ;                   execute the floating literal operation, writing the result
0030  7686 ;                   to the RDM's WHREG.
0030  7687 ;        8. Compare the WHREG against a table of expected data indexed by I.
0030  7688 ;        9. If the comparison shows that the two do not match, signal an error.
0030  7689 ;       10. Continue the loop started in step 3 for the remaining patterns.
0030  7690 ;       11. Stop signature analysis.
0030  7691 ;       12. End of the test, skip to the next test.
0030  7692 ;
0030  7693 ; TEST DATA
0030  7694 ;
0030  7695 ;       Six longword stuck at zero or one test patterns are input to the
0030  7696 ;       rotator from the MBUS with the RBUS held to zero. The floating point
0030  7697 ;       literal function extracts eight bits from the MBUS and places them onto
0030  7698 ;       the SBUS as follows:
0030  7699 ;
0030  7700 ;               SBUS <03:00> <-- 0
0030  7701 ;               SBUS <11:04> <-- MBUS<07:00>
0030  7702 ;               SBUS <13:12> <-- 0
0030  7703 ;               SBUS <14>    <-- 1
0030  7704 ;               SBUS <31:15> <-- 0
0030  7705 ;
0030  7706 ;       The input and output data patterns are:
0030  7707 ;
0030  7708 ;                 MBUS            SBUS
0030  7709 ;               AAAAAAAA        00004AA0
0030  7710 ;               55555555        00004550
0030  7711 ;               33333333        00004330
0030  7712 ;               0F0F0F0F        000040F0
```

```
                     0030  7713 ;                    00FF00FF        00004FF0
                     0030  7714 ;                    0000FFFF        00004FF0
                     0030  7715 ;
                     0030  7716 ; LOGIC DESCRIPTION
                     0030  7717 ;
                     0030  7718 ;       The floating literal extract is performed inside the SRM gate arrays.
                     0030  7719 ;       The SRM's are controlled by the SRK gate array via the control signals
                     0030  7720 ;       PRI<1:0> and SEC<5:0>.
                     0030  7721 ;
                     0030  7722 ; ASSUMPTIONS
                     0030  7723 ;
                     0030  7724 ;       This test assumes the following logic on the DPM has been tested and
                     0030  7725 ;       is working; the MBUS, RBUS, SBUS and WBUS; the scratchpad MTEMP0;
                     0030  7726 ;       and the ALPCTL data path for WX_S.Q_0.
                     0030  7727 ;
                     0030  7728 ; ERROR DESCRIPTION
                     0030  7729 ;
                     0030  7730 ;       On error this test will call out the SRM and SRK gate arrays. The
                     0030  7731 ;       following data will also by typed:
                     0030  7732 ;
                     0030  7733 ;       EXPECTED SBUS DATA
                     0030  7734 ;       RECEIVED SBUS DATA
                     0030  7735 ;       LOOP COUNT I
                     0030  7736 ;       MBUS DATA
                     0030  7737 ;--
                     0030  7738 ;*******************************************************************************
                     0030  7739 ;///////////////////////////////////////////////////////////////////////////////
                     0030  7740
                     0030  7741         INITIALIZE                                  ; INITIALIZE THE CPU
                     0032  7742         LOOP            I,1,6                       ; LOOP ON I FOR 6 SA01 PATTERNS
                     0039  7743         SA01PAT         I,1$                        ; GENERATE A SA01 PATTERN
                     0041  7744         LOADREG         WDREG,1$,,L                 ; LOAD WDREG WITH THE PATTERN
                     0049  7745         ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                     004B  7746         FETCH           0                           ; EXECUTE THE LITERAL EXPANSION
                     0050  7747         BRSTCLK         3                           ; AND WRITE THE RESULT TO WHREG
                     0054  7748         CMPREG          WHREG,2$,I,L                ; IS THE EXPANSION CORRECT?
                     005D  7749         IFERROR         1,<<SRM><SRK>>              ; IF NOT, SIGNAL AN ERROR
                     0064  7750         ENDLOOP         I                           ; CONTINUE LOOP FOR 6 PATTERNS
                     0067  7751         SKIP                                        ; END OF TEST
                     006E  7752
                     006E  7753 BEGIN_TEST_DATA
                     006E
                     006E       ;-------------------------------------------------------------------------
                     006E  7754
00000000             006E  7755 1$:     .LONG   0       ; LOCATION TO HOLD SA01 PATTERNS
                     0072  7756
                     0072  7757 ;
                     0072  7758 ; Expected Data
                     0072  7759 ;
                     0072  7760
00004AA0             0072  7761 2$:     .LONG   ^X00004AA0
00004550             0076  7762         .LONG   ^X00004550
00004330             007A  7763         .LONG   ^X00004330
000040F0             007E  7764         .LONG   ^X000040F0
00004FF0             0082  7765         .LONG   ^X00004FF0
```

```
                        00004FF0  0086  7766          .LONG   ^X00004FF0
                                  008A  7767
                                  008A  7768 BEGIN_CPU_DATA
                                  0002  7769
                                  0002  7770 DCS_DATA
                                  0001  7771
U 00, 7700,C800,0364,0300,0470,1801   0001  7772 ;% 00   NOP
U 01, 5700,C860,0364,0300,0470,1802   000C  7776 ;% 01   M[TEMP0]_RDM
U 02, 6700,0800,7B47,03D8,0470,1803   0017  7780 ;% 02   MSRC/TEMP0,RSRC/ZERO,ROT/FPLIT,ALPCTL/WX_S.Q_0,RDM_WB
U 03, 7300,C800,0364,0300,0470,1804   0022  7784 ;% 03   STOP.CLOCK
                                  002D  7788
                                  002D  7789 END_CPU_DATA
                                  008A  7790
                                  008A  7791 END_TEST_DATA
                                  008A
                                  008A       ;-------------------------------------------------------------------------
                                  008A  7792
                                  008A  7793 ENDTEST
```

```
0033        .SBTTL  "TEST   060     Rotator Test 46 Find Most Significant Bit Test
0033  7795 ;*******************************************************************************
0033  7796 ;++
0033  7797 ;
0033  7798 ; FUNCTIONAL DESCRIPTION
0033  7799 ;
0033  7800 ;       This test examines the rotator logic used to find the most significant
0033  7801 ;       bit set in a longword.
0033  7802 ;
0033  7803 ; TEST ALGORITHM
0033  7804 ;
0033  7805 ;       This test is implemented by floating a one across 32 bits, each time
0033  7806 ;       expecting the correct bit position for the one to be written into the
0033  7807 ;       PLATCH. The test algorithm executes as follows:
0033  7808 ;
0033  7809 ;        1. Initialize the CPU.
0033  7810 ;        2. Start signature analysis.
0033  7811 ;        3. Loop on I for 32 floating one patterns.
0033  7812 ;        4. Load the WDREG with a pattern from a table indexed by I.
0033  7813 ;        5. Loop on error from here.
0033  7814 ;        6. Execute the micro-code to:
0033  7815 ;                1. Write WDREG into MTEMP0 and RTEMP0.
0033  7816 ;                2. Source RTEMP0 onto the RBUS, MTEMP0 onto the MBUS, route
0033  7817 ;                   the RBUS onto the WBUS using the ALPCTL micro-order WX_R.Q_D
0033  7818 ;                   and execute the find most significant bit function.
0033  7819 ;                3. Write the contents of the PLATCH into WHREG.
0033  7820 ;        7. Compare WHREG against a table of expected data indexed by I.
0033  7821 ;        8. If the two do not match, signal an error.
0033  7822 ;        9. Continue the loop from step 3 until all patterns have been tried.
0033  7823 ;       10. Stop signature analysis.
0033  7824 ;       11. End of the test, skip to the next test.
0033  7825 ;
0033  7826 ; TEST DATA
0033  7827 ;
0033  7828 ;       This test uses 32 longword floating one patterns for input data and
0033  7829 ;       32 bytes of decimal data for expected output data.
0033  7830 ;       The data is as follows:
0033  7831 ;
0033  7832 ;          MBUS RBUS WBUS Data       Expected PLATCH contents
0033  7833 ;               00000001 (hex)                 00 (decimal)
0033  7834 ;               00000002                       01
0033  7835 ;               00000004                       02
0033  7836 ;               00000008                       03
0033  7837 ;               00000010                       04
0033  7838 ;               00000020                       05
0033  7839 ;               00000040                       06
0033  7840 ;               00000080                       07
0033  7841 ;               00000100                       08
0033  7842 ;               00000200                       09
0033  7843 ;               00000400                       10
0033  7844 ;               00000800                       11
0033  7845 ;               00001000                       12
0033  7846 ;               00002000                       13
0033  7847 ;               00004000                       14
0033  7848 ;               00008000                       15
```

```
0033  ;                 00010000                        16
0033  ;                 00020000                        17
0033  ;                 00040000                        18
0033  ;                 00080000                        19
0033  ;                 00100000                        20
0033  ;                 00200000                        21
0033  ;                 00400000                        22
0033  ;                 00800000                        23
0033  ;                 01000000                        24
0033  ;                 02000000                        25
0033  ;                 04000000                        26
0033  ;                 08000000                        27
0033  ;                 10000000                        28
0033  ;                 20000000                        29
0033  ;                 40000000                        30
0033  ;                 80000000                        31
0033  ;
0033  ; LOGIC DESCRIPTION
0033  ;
0033  ;       The find most significant bit logic is inside the SRK gate array.
0033  ;
0033  ; ASSUMPTIONS
0033  ;
0033  ;       This test assumes the following logic on the DPM has been tested and
0033  ;       is working; the MBUS, RBUS, SBUS and WBUS; the scratchpads MTEMP0 and
0033  ;       RTEMP0; and the ALPCTL data path for WX_R.Q_D.
0033  ;
0033  ; ERROR DESCRIPTION
0033  ;
0033  ;       On error this test will call out the SRK gate array.
0033  ;       The following data will also by typed:
0033  ;
0033  ;       EXPECTED PLATCH CONTENTS
0033  ;       RECEIVED PLATCH CONTENTS
0033  ;       LOOP COUNT I
0033  ;       MBUS, RBUS AND WBUS DATA
0033  ;--
0033  ;*******************************************************************************
0033  ;///////////////////////////////////////////////////////////////////////////////
0033
0033          INITIALIZE                                  ; INITIALIZE THE CPU
0035          LOOP            I,1,32                      ; LOOP FOR 32 TEST PATTERNS
003C          LOADREG         WDREG,1$,I,L                ; LOAD A FLOATING 1 PATTERN
0044          ERRLOOP                                     ; LOOP ON ERROR FROM HERE
0046          FETCH           0                           ; EXECUTE THE ROT/PL_MSS TEST
004B          BRSTCLK         4                           ; AND WRITE RESULT TO WHREG
004F          CMPREG          WHREG,2$,I,B                ; WAS THE PLATCH CORRECT?
0058          IFERROR         1,<SRK>                     ; IF NOT, SIGNAL AN ERROR
005E          ENDLOOP         I                           ; CONTINUE LOOP FOR 32 PATTERNS
0061          SKIP                                        ; END OF TEST
0068
0068  BEGIN_TEST_DATA
0068
0068      ;-----------------------------------------------------------------------
0068
```

```
                                0068  7902 ;
                                0068  7903 ; MBUS Input Data
                                0068  7904 ;
                                0068  7905
00000004 00000002 00000001      0068  7906 1$:     .LONG   ^X00000001,^X00000002,^X00000004        ; I = 01 - 03
00000020 00000010 00000008      0074  7907         .LONG   ^X00000008,^X00000010,^X00000020        ; I = 04 - 06
00000100 00000080 00000040      0080  7908         .LONG   ^X00000040,^X00000080,^X00000100        ; I = 07 - 09
00000800 00000400 00000200      008C  7909         .LONG   ^X00000200,^X00000400,^X00000800        ; I = 0A - 0C
00004000 00002000 00001000      0098  7910         .LONG   ^X00001000,^X00002000,^X00004000        ; I = 0D - 0F
00020000 00010000 00008000      00A4  7911         .LONG   ^X00008000,^X00010000,^X00020000        ; I = 10 - 12
00100000 00080000 00040000      00B0  7912         .LONG   ^X00040000,^X00080000,^X00100000        ; I = 13 - 15
00800000 00400000 00200000      00BC  7913         .LONG   ^X00200000,^X00400000,^X00800000        ; I = 16 - 18
04000000 02000000 01000000      00C8  7914         .LONG   ^X01000000,^X02000000,^X04000000        ; I = 19 - 1B
20000000 10000000 08000000      00D4  7915         .LONG   ^X08000000,^X10000000,^X20000000        ; I = 1C - 1E
         80000000 40000000      00E0  7916         .LONG   ^X40000000,^X80000000                   ; I = 1F - 20
                                00E8  7917
                                00E8  7918 ;
                                00E8  7919 ; Expected PLATCH Data
                                00E8  7920 ;
                                00E8  7921
   07 06 05 04 03 02 01 00      00E8  7922 2$:     .BYTE   00,01,02,03,04,05,06,07                 ; I = 01 - 08
   0F 0E 0D 0C 0B 0A 09 08      00F0  7923         .BYTE   08,09,10,11,12,13,14,15                 ; I = 09 - 10
   17 16 15 14 13 12 11 10      00F8  7924         .BYTE   16,17,18,19,20,21,22,23                 ; I = 11 - 18
   1F 1E 1D 1C 1B 1A 19 18      0100  7925         .BYTE   24,25,26,27,28,29,30,31                 ; I = 19 - 20
                                0108  7926
                                0108  7927 BEGIN_CPU_DATA
                                0002  7928
                                0002  7929 DCS_DATA
                                0001  7930
U 00, 7700,C800,0364,0300,0470,1801  0001  7931 ;% 00    NOP
U 01, 5700,C860,0364,0300,0470,1802  000C  7935 ;% 01    M[TEMP0]_RDM
U 02, 7700,4800,9E47,0300,0470,1803  0017  7939 ;% 02    RSRC/TEMP0,MSRC/TEMP0,ALPCTL/WX_R.Q_D,ROT/PL_MSS
U 03, 6700,C800,B347,0300,0470,1804  0022  7943 ;% 03    ROT/PL,ALPCTL/WX_S.Q_0,RDM_WB
U 04, 7300,4800,0364,0300,0470,1805  002D  7947 ;% 04    STOP.CLOCK
                                0038  7951
                                0038  7952 END_CPU_DATA
                                0108  7953
                                0108  7954 END_TEST_DATA
                                0108
                                0108      ;-------------------------------------------------------------------------------
                                0108  7955
                                0108  7956 ENDTEST
```

```
                        0022        .SBTTL  "TEST   061     Rotator Test 47 BCD Swap Test
                        0022  7958 ;*******************************************************************************
                        0022  7959 ;++
                        0022  7960 ;
                        0022  7961 ; FUNCTIONAL DESCRIPTION
                        0022  7962 ;
                        0022  7963 ;       This test will examine the rotator's ability to perform a BCD byte swap
                        0022  7964 ;       of data on the MBUS.
                        0022  7965 ;
                        0022  7966 ; TEST ALGORITHM
                        0022  7967 ;
                        0022  7968 ;       This test is implemented by the following algorithm:
                        0022  7969 ;
                        0022  7970 ;        1. Initialize the CPU.
                        0022  7971 ;        2. Start signature analysis.
                        0022  7972 ;        3. Loop on I for six longword stuck at zero or one test patterns.
                        0022  7973 ;        4. Generate a stuck at zero or one pattern based on I.
                        0022  7974 ;        5. Load the pattern into the WDREG.
                        0022  7975 ;        6. Loop on error from here.
                        0022  7976 ;        7. Execute the micro-code to:
                        0022  7977 ;                1. Write the WDREG to MTEMP0.
                        0022  7978 ;                2. Source zero onto the RBUS, MTEMP0 onto the MBUS and execute
                        0022  7979 ;                   the BCD byte swap, routing the SBUS result through the ALP
                        0022  7980 ;                   to the WBUS and the RDM's WHREG.
                        0022  7981 ;        8. Compare the WHREG against a table of expected data indexed by I.
                        0022  7982 ;        9. If the two do not match, signal an error.
                        0022  7983 ;       10. Loop back to step 3 until all test patterns have been tried.
                        0022  7984 ;       11. Stop signature analysis.
                        0022  7985 ;       12. End of the test, skip to the next test.
                        0022  7986 ;
                        0022  7987 ; TEST DATA
                        0022  7988 ;
                        0022  7989 ;       For this test bytes on the MBUS are swapped according to the following
                        0022  7990 ;       format:
                        0022  7991 ;
                        0022  7992 ;               SBUS <07:00> <-- MBUS <31:24>
                        0022  7993 ;               SBUS <15:08> <-- MBUS <23:16>
                        0022  7994 ;               SBUS <23:16> <-- MBUS <15:08>
                        0022  7995 ;               SBUS <31:24> <-- MBUS <07:00>
                        0022  7996 ;
                        0022  7997 ;       Six longword stuck at zero or one test patterns are placed on the
                        0022  7998 ;       MBUS with constants zeroes on the RBUS. The input data and the expected
                        0022  7999 ;       output data are as follows:
                        0022  8000 ;
                        0022  8001 ;                 MBUS            SBUS
                        0022  8002 ;               AAAAAAAA        AAAAAAAA
                        0022  8003 ;               55555555        55555555
                        0022  8004 ;               33333333        33333333
                        0022  8005 ;               0F0F0F0F        0F0F0F0F
                        0022  8006 ;               00FF00FF        FF00FF00
                        0022  8007 ;               0000FFFF        FFFF0000
                        0022  8008 ;
                        0022  8009 ; LOGIC DESCRIPTION
                        0022  8010 ;
                        0022  8011 ;       The BCD swap function is carried out by the SRM gate arrays which are
```

```
                        0022  8012 ;       controlled by the SRK gate array via the control signals PRI<1:0> and
                        0022  8013 ;       SEC<5:0>.
                        0022  8014 ;
                        0022  8015 ; ASSUMPTIONS
                        0022  8016 ;
                        0022  8017 ;       This test assumes the following logic has been tested and is working:
                        0022  8018 ;       the RBUS, MBUS, SBUS and WBUS; the scratchpad MTEMP0; and the ALPCTL
                        0022  8019 ;       data path WX_S.Q_0.
                        0022  8020 ;
                        0022  8021 ; ERROR DESCRIPTION
                        0022  8022 ;
                        0022  8023 ;       On error this test will call out the SRM and SRK gate arrays.
                        0022  8024 ;       It will also type out the following data:
                        0022  8025 ;
                        0022  8026 ;       EXPECTED SBUS DATA
                        0022  8027 ;       RECEIVED SBUS DATA
                        0022  8028 ;       LOOP COUNT I
                        0022  8029 ;       MBUS DATA
                        0022  8030 ;--
                        0022  8031 ;*******************************************************************************
                        0022  8032 ;///////////////////////////////////////////////////////////////////////////////
                        0022  8033
                        0022  8034         INITIALIZE                              ; INITIALIZE THE CPU
                        0024  8035         LOOP            I,1,6                   ; LOOP ON I FOR 6 SA01 PATTERNS
                        002B  8036         SA01PAT         I,1$                    ; GENERATE A SA01 PATTERN
                        0033  8037         LOADREG         WDREG,1$,,L             ; LOAD WDREG WITH SA01 PATTERN
                        003B  8038         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        003D  8039         FETCH           0                       ; EXECUTE THE BCD SWAP AND
                        0042  8040         BRSTCLK         3                       ; WRITE THE RESULT TO WHREG
                        0046  8041         CMPREG          WHREG,2$,I,L            ; IS THE SWAP CORRECT?
                        004F  8042         IFERROR         1,<SRK>                 ; IF NOT, SIGNAL AN ERROR
                        0055  8043         ENDLOOP         I                       ; CONTINUE LOOP FOR 6 PATTERNS
                        0058  8044         SKIP                                    ; END OF TEST
                        005F  8045
                        005F  8046 BEGIN_TEST_DATA
                        005F
                        005F               ;-------------------------------------------------------------------------
                        005F  8047
00000000                005F  8048 1$:      .LONG   0       ; TEMPORARY TO HOLD SA01 PATTERN
                        0063  8049
                        0063  8050 ;
                        0063  8051 ; Expected SBUS Data
                        0063  8052 ;
                        0063  8053
AAAAAAAA                0063  8054 2$:      .LONG   ^XAAAAAAAA
55555555                0067  8055          .LONG   ^X55555555
33333333                006B  8056          .LONG   ^X33333333
0F0F0F0F                006F  8057          .LONG   ^X0F0F0F0F
FF00FF00                0073  8058          .LONG   ^XFF00FF00
FFFF0000                0077  8059          .LONG   ^XFFFF0000
                        007B  8060
                        007B  8061 BEGIN_CPU_DATA
                        0002  8062
                        0002  8063 DCS_DATA
                        0001  8064
```

```
ZZ-ECKAB-9.2    "TEST 061 Rotator Test 47 BCD Swap Test              13-MAY-1987       Fiche 3 Frame E3          Sequence 442
ECKAB                          VAX-11/750 DPM MICRO DIAGNOSTICS        11-DEC-1985 09:34:57  VAX/VMS Macro V04-00        Page 17
V09.01                         "TEST 061 Rotator Test 47 BCD Swap Test  11-DEC-1985 09:34:14  [VAX750.DPM]ECKAB2.MAR;1          (1

U 00, 7700,C800,0364,0300,0470,1801   0001  8065 ;% 00    NOP
U 01, 5700,C860,0364,0300,0470,1802   000C  8069 ;% 01    M[TEMP0]_RDM
U 02, 6700,0800,6347,0300,0470,1803   0017  8073 ;% 02    MSRC/TEMP0,ROT/BCDSWP,ALPCTL/WX_S.Q_0,RDM_WB
U 03, 7300,C800,0364,0300,0470,1804   0022  8077 ;% 03    STOP.CLOCK
                                      002D  8081
                                      002D  8082 END_CPU_DATA
                                      007B  8083
                                      007B  8084 END_TEST_DATA
                                      007B
                                      007B       ;-----------------------------------------------------------------------------
                                      007B  8085
                                      007B  8086 ENDTEST
```

```
0030        .SBTTL  "TEST   062     Rotator Test 48 Convert Numeric to BCD Test
0030  8088 ;*******************************************************************************
0030  8089 ;++
0030  8090 ;
0030  8091 ; FUNCTIONAL DESCRIPTION
0030  8092 ;
0030  8093 ;       This test examines the rotator's ability to covert eight numeric digits
0030  8094 ;       on the MBUS concatenated with the RBUS to eight BCD digits.
0030  8095 ;
0030  8096 ; TEST ALGORITHM
0030  8097 ;
0030  8098 ;       This test is implemented using seven 64 bit stuck at zero or one test
0030  8099 ;       patterns. These patterns are input to the rotator from the MBUS
0030  8100 ;       concatenated with the RBUS. They are then extracted and shuffled by
0030  8101 ;       the convert function before being placed on the SBUS. From the SBUS
0030  8102 ;       they are passed through the ALP's to the WBUS and the RDM's WHREG.
0030  8103 ;       The test algorithm executes as follows:
0030  8104 ;
0030  8105 ;        1. Initialize the CPU.
0030  8106 ;        2. Start signature analysis.
0030  8107 ;        3. Load the micro-code for the first six test patterns.
0030  8108 ;        4. Loop on I for 6 stuck at zero or one test patterns.
0030  8109 ;        5. Generate a stuck a zero or one test pattern for a longword.
0030  8110 ;        6. Load the pattern into the WDREG.
0030  8111 ;        7. Loop on error from here.
0030  8112 ;        8. Execute the micro-code to:
0030  8113 ;                1. Load the test pattern from WDREG to MTEMP0.
0030  8114 ;                2. Load the test pattern from WDREG to RTEMP1.
0030  8115 ;                3. Placing RTEMP1 on the RBUS and MTEMP0 on the MBUS, execute
0030  8116 ;                   the convert function and route the result on the SBUS
0030  8117 ;                   through the ALP's WMUX to the RDM's WHREG.
0030  8118 ;        9. Compare the contents of the WHREG against a table of expected data.
0030  8119 ;       10. If the two do not match, signal an error.
0030  8120 ;       11. Continue the loop started at step 4 for the remaining patterns.
0030  8121 ;       12. Load the micro-code for the seventh test pattern.
0030  8122 ;       13. Loop once on I = 7 for the last stuck at zero or one test pattern.
0030  8123 ;       14. Loop on error from here.
0030  8124 ;       15. Execute the micro-code to:
0030  8125 ;                4. Load the LONLIT register with all ones.
0030  8126 ;                5. Load MTEMP0 with zero.
0030  8127 ;                6. Load MTEMP1 (and RTEMP1) with the LONLIT register.
0030  8128 ;                7. Placing RTEMP1 on the RBUS and MTEMP0 on the MBUS, execute
0030  8129 ;                   the convert function and route the result on the SBUS
0030  8130 ;                   through the ALP's WMUX to the RDM's WHREG.
0030  8131 ;       16. Compare the contents of the WHREG against a table of expected data.
0030  8132 ;       17. If the two do not match, signal an error.
0030  8133 ;       18. End the loop started at step 13.
0030  8134 ;       19. Stop signature analysis.
0030  8135 ;       20. End of this test, skip to the next test.
0030  8136 ;
0030  8137 ; TEST DATA
0030  8138 ;
0030  8139 ;       Seven stuck at zero or one test patterns are used to test the numeric
0030  8140 ;       to BCD conversion. The conversion extracts and merges fields from the
0030  8141 ;       MBUS concatenated with the RBUS and places the result on the SBUS as
```

```
0030  8142 ;       follows:
0030  8143 ;
0030  8144 ;               SBUS <03:00> <-- MBUS <11:08>
0030  8145 ;               SBUS <07:04> <-- MBUS <03:00>
0030  8146 ;               SBUS <11:08> <-- MBUS <27:24>
0030  8147 ;               SBUS <15:12> <-- MBUS <19:16>
0030  8148 ;               SBUS <19:16> <-- RBUS <11:08>
0030  8149 ;               SBUS <23:20> <-- RBUS <03:00>
0030  8150 ;               SBUS <27:24> <-- RBUS <27:24>
0030  8151 ;               SBUS <31:28> <-- RBUS <19:16>
0030  8152 ;
0030  8153 ;       The input and output data patterns used are:
0030  8154 ;
0030  8155 ;                   Input Data      Expected Output Data
0030  8156 ;
0030  8157 ;                 MBUS     RBUS          SBUS
0030  8158 ;                AAAAAAAA AAAAAAAA       AAAAAAAA
0030  8159 ;                55555555 55555555       55555555
0030  8160 ;                33333333 33333333       33333333
0030  8161 ;                0F0F0F0F 0F0F0F0F       FFFFFFFF
0030  8162 ;                00FF00FF 00FF00FF       F0F0F0F0
0030  8163 ;                0000FFFF 0000FFFF       00FF00FF
0030  8164 ;                00000000 FFFFFFFF       FFFF0000
0030  8165 ;
0030  8166 ; LOGIC DESCRIPTION
0030  8167 ;
0030  8168 ;       The numeric to BCD conversion is performed inside the SRM gate arrays.
0030  8169 ;       The SRM's are controlled by the SRK gate array via the control signals
0030  8170 ;       PRI<1:0> and SEC<5:0>.
0030  8171 ;
0030  8172 ; ASSUMPTIONS
0030  8173 ;
0030  8174 ;       This test assumes the following logic on the DPM has been tested and
0030  8175 ;       is working; the MBUS, RBUS, SBUS and WBUS; the scratchpads RTEMP1 and
0030  8176 ;       MTEMP0; and the ALPCTL data path for WX_S.Q_0.
0030  8177 ;
0030  8178 ; ERROR DESCRIPTION
0030  8179 ;
0030  8180 ;       On error this test will call out the SRM and SRK gate arrays. The
0030  8181 ;       following data will also by typed:
0030  8182 ;
0030  8183 ;       EXPECTED SBUS DATA
0030  8184 ;       RECEIVED SBUS DATA
0030  8185 ;       LOOP COUNT I
0030  8186 ;       MBUS DATA
0030  8187 ;       RBUS DATA
0030  8188 ;--
0030  8189 ;*******************************************************************************
0030  8190 ;///////////////////////////////////////////////////////////////////////////////
0030  8191
0030  8192         INITIALIZE                                      ; INITIALIZE THE CPU
0032  8193         LOADDCS         10$.,0.5                        ; LOAD FIRST MICRO-CODE SET
0039  8194         LOOP            I,1,6                           ; LOOP ON I FOR 6 SA01 PATTERNS
0040  8195         SA01PAT         I,1$                            ; GENERATE A SA01 PATTERN
0048  8196         LOADREG         WDREG,1$,,L                     ; LOAD WDREG WITH PATTERN
```

```
                        0050  8197          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        0052  8198          FETCH           0                       ; EXECUTE THE COVERSION AND
                        0057  8199          BRSTCLK         4                       ; WRITE THE RESULT TO WHREG
                        005B  8200          CMPREG          WHREG,2$,I,L            ; IS THE CONVERSION CORRECT?
                        0064  8201          IFERROR         2,<<SRM><SRK>>          ; IF NOT, SIGNAL AN ERROR
                        006B  8202          ENDLOOP         I                       ; CONTINUE LOOP
                        006E  8203          LOADDCS         20$.,0,6                ; LOAD SECOND MICRO-CODE SET
                        0075  8204          LOOP            I,7,7                   ; LOOP ONCE FOR 7TH PATTERN
                        007C  8205          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        007E  8206          FETCH           0                       ; EXECUTE THE CONVERSION ON 7TH
                        0083  8207          BRSTCLK         5                       ; PATTERN AND WRITE TO WHREG
                        0087  8208          CMPREG          WHREG,2$,I,L            ; IS THE CONVERSION CORRECT?
                        0090  8209          IFERROR         2,<<SRM><SRK>>          ; IF NOT, SIGNAL AN ERROR
                        0097  8210          ENDLOOP         I                       ; END OF LOOP FOR 7TH PATTERN
                        009A  8211          SKIP                                    ; END OF TEST
                        00A1  8212
                        00A1  8213 BEGIN_TEST_DATA
                        00A1
                        00A1          ;-------------------------------------------------------------------------
                        00A1  8214
00000000                00A1  8215 1$:      .LONG   0       ; LOCATION TO HOLD SA01 PATTERNS
                        00A5  8216
                        00A5  8217 ;
                        00A5  8218 ; Expected Data
                        00A5  8219 ;
                        00A5  8220
AAAAAAAA                00A5  8221 2$:      .LONG   ^XAAAAAAAA
55555555                00A9  8222          .LONG   ^X55555555
33333333                00AD  8223          .LONG   ^X33333333
FFFFFFFF                00B1  8224          .LONG   ^XFFFFFFFF
F0F0F0F0                00B5  8225          .LONG   ^XF0F0F0F0
00FF00FF                00B9  8226          .LONG   ^X00FF00FF
FFFF0000                00BD  8227          .LONG   ^XFFFF0000
                        00C1  8228
                        00C1  8229 ;
                        00C1  8230 ; Test micro-code sets
                        00C1  8231 ;
                        00C1  8232
                        00C1  8233 10$:
U 00, 7700,C800,0364,0300,0470,1801     00C1  8234 ;% 00    NOP
U 01, 5700,C860,0364,0300,0470,1802     00CC  8238 ;% 01    M[TEMP0]_RDM
U 02, 5700,4840,0364,0304,0470,1803     00D7  8242 ;% 02    R[TEMP1]_RDM
U 03, 6700,8800,7F47,0304,0470,1804     00E2  8246 ;% 03    RSRC/TEMP1,MSRC/TEMP0,ROT/CVTNP,ALPCTL/WX_S.Q_0,RDM_WB
U 04, 7300,4800,0364,0300,0470,1805     00ED  8250 ;% 04    STOP.CLOCK
                        00F8  8254
                        00F8  8255 20$:
U 00, 7700,C800,0364,0300,0470,1801     00F8  8256 ;% 00    NOP
U 01, 7700,B800,0000,0000,0470,1802     0103  8260 ;% 01    LONLIT_[0FFFFFFFF]
U 02, 7700,4860,5BE4,03D8,0470,1803     010E  8264 ;% 02    M[TEMP0]_R[ZERO]
U 03, 7700,8861,5BE4,03D4,0470,1804     0119  8268 ;% 03    M[TEMP1]_R[LONLIT]
U 04, 6700,0800,7F47,0304,0470,1805     0124  8272 ;% 04    RSRC/TEMP1,MSRC/TEMP0,ROT/CVTNP,ALPCTL/WX_S.Q_0,RDM_WB
U 05, 7300,4800,0364,0300,0470,1806     012F  8276 ;% 05    STOP.CLOCK
                        013A  8280
                        013A  8281 END_TEST_DATA
                        013A
```

```
013A        ;-----------------------------------------------------------------------
013A  8282
013A  8283 ENDTEST
```

```
                        0000  8285 .END
```

```
$CONTROL_C_CODE= 0000000A
$FILE_NOT_FOUND= 00000009
$STN            = 00000000
$TEST_NUMBER    = 00000063
ACV             = 0000000F
AC_LOW          = 00000080
ADD             = 00000019
ADDR_MATCH_HI   = 00007405
ADDR_MATCH_LO   = 00007404
ADK             = 0000000B
ALK             = 00000000
ALP             = 00000016
ARG_CNT         = 00000091 R      FD
A_REG_BYTE_0    = 00007410
A_REG_BYTE_1    = 00007411
A_REG_BYTE_2    = 00007412
A_REG_BYTE_3    = 00007413
B               = 00000001
BADD1           = 00007426
BELL_FLAG       = 00000008
BELL_KEY        = 0000000E
BUFF_ADDR_LOW   = 00007426
BURST_STOP_FLAG= 00000080
BYTE            = 00000001
CACHE_FLAG      = 00000000
CACHE_FLAG1     = 00000000
CACHE_LENGTH    = 00000001
CAK             = 0000000A
CCC             = 00000004
CCS             = 00000005
CF_KEY          = 00000024
CLA             = 00000003
CLEAR_CTRL_FILE= 00000002
CLKDCS          = 00007427
CLK_CTRL_0      = 00000020
CLK_CTRL_0_RO   = 00000040
CLK_CTRL_1      = 00000040
CLK_CTRL_1_RO   = 00000080
CLOCK_STOPED    = 00000080
CMI             = 00000007
CMICTL          = 0000741C
CMI_COMPLETE    = 00000020
CMI_CTRL_CLEAR  = 00000001
CMI_CTRL_REG    = 0000741C
CMI_GO          = 00000008
CMI_READ        = 00000040
CMI_STATUS_0    = 00000008
CMI_STATUS_1    = 00000010
CMI_WRITE       = 00000020
CML             = 0000000E
COMPOSIT_LENGTH= 00000003
CON             = 00000011
CONTINUE_FLAG   = 00000002
CONTROL_C_FLAG  = 00000001
CONT_FLAG       = 00000020
CONT_KEY        = 0000001C
CPU_RUN         = 00000010
CSAMR           = 00007404
CSBUF           = 00007424
CSTRP           = 00007422
CS_ADD_BUFF_HGH= 00007425
CS_ADD_BUFF_LOW= 00007424
CS_ADD_TRAP_HGH= 00007423
CS_ADD_TRAP_LOW= 00007422
CYCLE_FLAG      = 00000008
CYCLE_KEY       = 00000020
D0              = 00000002
D1              = 00000006
D2              = 00000000
DATA_FLAG       = 00000000
DATA_LENGTH     = 00000000
DCSAD           = 00007401
DCSCF           = 00007403
DCSCR           = 00007402
DCSDA           = 00007400
DCS_ADDR_CLEAR  = 00000002
DCS_ADDR_REG_RO= 00007427
DCS_ADDR_REG_WO= 00007401
DCS_CTRL_FILE   = 00007403
DCS_CTRL_REG    = 00007402
DCS_DATA_REG    = 00007400
DCS_FLAG        = 00000000
DCS_FLAG1       = 00000000
DCS_HEAD        = 00000000 R      F8
DCS_LENGTH      = 00000001
DCS_SCRATCH_ADR= 00000036
DCS_START       = 00001800
DC_LOW          = 00000040
DIAGNOSE_FLAG   = 00000004
DPM             = 00000000
DUM1            = FFFFFFF2
ENDTEST_LOC     = 0000013F R      FD
END_SLICE       = 00000025
END_TEST_55       00000090 R      02
END_TEST_56       00000090 R      09
END_TEST_57       00000092 R      0F
END_TEST_58       00000092 R      15
END_TEST_59       0000008B R      1B
END_TEST_60       00000090 R      21
END_TEST_61       000000D7 R      27
END_TEST_62       00000139 R      2D
END_TEST_63       00000233 R      33
END_TEST_64       00000262 R      39
END_TEST_65       00000262 R      3F
END_TEST_66       00000087 R      45
END_TEST_67       0000008F R      4B
END_TEST_68       00000082 R      51
END_TEST_69       00000084 R      57
END_TEST_70       0000015D R      5D
END_TEST_71       000000FA R      63
END_TEST_72       0000019F R      69
END_TEST_73       00000232 R      6F
END_TEST_74       000002A1 R      75
END_TEST_75       00000232 R      7B
END_TEST_76       000002A1 R      81
END_TEST_77       0000022F R      87
END_TEST_78       000002BF R      8D
END_TEST_79       00000445 R      93
END_TEST_80       00000281 R      99
END_TEST_81       0000018D R      9F
END_TEST_82       000002F1 R      A5
END_TEST_83       00000319 R      AB
END_TEST_84       00000152 R      B1
END_TEST_85       000000FD R      B7
END_TEST_86       00000270 R      BD
END_TEST_87       00000117 R      C3
END_TEST_88       00000064 R      C9
END_TEST_89       0000011B R      CF
END_TEST_90       0000011B R      D5
END_TEST_91       00000102 R      DB
END_TEST_92       00000095 R      E1
END_TEST_93       00000142 R      E7
END_TEST_94       0000013E R      E9
END_TEST_95       0000008A R      EB
END_TEST_96       00000108 R      F1
END_TEST_97       0000007B R      F7
END_TEST_98       0000013A R      FD
EN_TRACE_REG    = 00000008
ERRLOG_FLAG     = 00000000
ERROR_FLAG      = 00000010
ERROR_LOG_ADR   = 000032FC
ERR_LOG_INIT    = 00000001
EXP_STALL_FLAG  = 00000010
EXP_UTRAP_FLAG  = 00000040
FAC             = 00000020
FAULT           = 00000002
FCC             = 00000015
FCS             = 00000021
FETCH_FLAG      = 00000008
FEX             = 0000001E
FFH             = 00000024
FFL             = 00000023
FIC             = 0000001F
FIO             = 0000001D
FLAG_KEY        = 00000004
FMR             = 00000022
FPA             = 00000002
FP_BOOT_0       = 00000001
FP_BOOT_1       = 00000002
FP_START_0      = 00000004
FP_START_1      = 00000008
FQA             = 00000014
FRONT1          = 00007420
FRONT2          = 00007421
FRONT_PNL_1     = 00007420
```

```
FRONT_PNL_2     = 00007421          MDL             = 0000001B          OP_SUB_TEST     = 0000001E
FRONT_PNL_LOCK  = 00000080          MDR             = 00000018          OVERLAY_HEAD    = 00000000 R     FE
HALT_FLAG       = 00000001          MDREG           = 00007414          PARITY_ERROR    = 00000020
HALT_KEY        = 00000008          MD_KEY          = 00000034          PAR_CHK_ENABL   = 00000008
HALT_ON_MATCH   = 00000001          MD_REG_BYTE_0   = 00007414          PAR_STOP_ENABL  = 00000010
HEAD            = FFFF7C00          MD_REG_BYTE_1   = 00007415          PASS_KEY        = 00000002
HIGH            = 00000001          MD_REG_BYTE_2   = 00007416          PB_INIT         = 00000040
H_FROM_WBUS     = 00000010          MD_REG_BYTE_3   = 00007417          PB_KEY          = 00000036
IB_FLAG         = 00000020          MEC             = 0000001A          PC_KEY          = 00000030
IB_KEY          = 00000012          MHREG           = 00007418          PHB             = 00000007
INSTR_FLAG      = 00000004          MH_REG_BYTE_0   = 00007418          PRK             = 0000000C
INSTR_KEY       = 00000018          MH_REG_BYTE_1   = 00007419          PS_KEY          = 0000003E
INT             = 00000010          MH_REG_BYTE_2   = 0000741A          QA_FLAG         = 00000040
IRD             = 00000005          MH_REG_BYTE_3   = 0000741B          QA_KEY          = 00000014
I_INDEX_FLAG    = 00000000          MIC             = 00000001          QV_FLAG         = 00000002
I_INIT          = 00000000          MICROWORD       = 0000000A          QV_KEY          = 00000028
J_INDEX_FLAG    = 00000000          MICRO_ADDR_INH  = 00000080          RDCTRL          = 0000741E
J_INIT          = 00000000          MONITOR_SIZE    = 00002C5E          RDM             = 00000004
K_INDEX_FLAG    = 00000000          MSQ             = 00000008          RDM_FLAG        = 00000004
K_INIT          = 00000000          MT_KEY          = 0000002E          RDM_KEY         = 0000002A
L               = 00000004          N               = 00000002          RDM_LAST        = 00000008
LIST_END        = 00000097 R     FD NER_FLAG        = 00000004          RD_CTRL_REG     = 0000741E
LIST_HEAD       = 00000092 R     FD NER_KEY         = 0000000C          REMOTE          = 00000001
LIST_LENGTH     = 00000005          NOERROR         = 00000001          RTEMP_FLAG      = 00000000
LITERAL         = 00000000          NOTEQUAL        = 00000003          RTEMP_FLAG1     = 00000000
LONG            = 00000004          ONEQUAL         = 00000002          RTEMP_HEAD      = 00000000 R     E3
LOOP_ERROR_FLAG= 00000020           ONERROR         = 00000000          RTEMP_LENGTH    = 00000001
LOOP_FLAG       = 00000002          OP_BEGIN_LOOP   = 00000008          RT_KEY          = 0000002C
LOOP_KEY        = 0000000A          OP_BURST_CLOCK  = 00000002          SAC             = 00000009
LOST_FLAG       = 00000010          OP_COMPARE_REG  = 00000004          SA_CLOCK        = 00000080
LOST_KEY        = 0000001A          OP_COMPARE_REGM= 00000006           SA_FLAG         = 00000010
LOW             = 00000000          OP_COMPARE_VB   = 00000020          SA_KEY          = 00000010
M               = 0000000A          OP_DUMPLOG      = 00000030          SA_START_STOP   = 00000080
M$TEST_SIZE     = 000007A2          OP_ENABL_STALL  = 00000038          SBE_FLAG        = 00000020
M$TEST_SIZE_D   = 000003E2          OP_ENABL_TRAP   = 0000002E          SBE_KEY         = 00000042
MA0             = 00000008          OP_END_FILE     = 00000024          SECTION_FLAG    = 00000080
MA1             = 0000000A          OP_END_LOOP     = 0000000A          SF_KEY          = 00000040
MAD             = 00000013          OP_END_TEST     = 0000000C          SOMM_FLAG       = 00000001
MAIN32          = 0000741D          OP_ERRLOG       = 00000032          SOMM_KEY        = 00000006
MAINT_A_TO_CMI  = 00000080          OP_ERROR_LOOP   = 0000000E          SPA             = 00000002
MAINT_CMI_TO_MH= 00000020           OP_FETCH        = 00000022          SRK             = 00000001
MAINT_DCS_ENABL= 00000004           OP_IF_ERROR     = 00000012          SRM             = 00000017
MAINT_MD_TO_CMI= 00000040           OP_INC_INDEX    = 0000002A          SR_KEY          = 0000003C
MAINT_REG       = 0000741D          OP_INIT         = 00000014          STACK_SIZE      = 00000400
MAINT_STB_INH   = 00000002          OP_LOAD_DCS     = 00000000          STATUS          = 00007406
MAINT_TRAP_OFF  = 00000001          OP_LOAD_FIELD   = 00000026          STATUS_REG      = 00007406
MAINT_WB_TO_WH  = 00000008          OP_LOAD_INDEX   = 00000028          STEP_KEY        = 0000001E
MAINT_WD_TO_WB  = 00000010          OP_LOAD_REG     = 00000016          STOP_CLOCK      = 00000004
MAP             = 00000012          OP_MASK         = 00000018          STROBE_CMI      = 00000040
MAREG           = 00007410          OP_NEW_TEST     = 0000001A          ST_KEY          = 0000001E
MASTER_HALT_EN  = 00000008          OP_PATTERN_GEN  = 00000010          TEMP            = 0000017E R     FD
MAX_ARGUMENTS   = 00000010          OP_PRINT_N      = 00000036          TEMP1           = 00000003
MA_KEY          = 00000038          OP_PRINT_S      = 00000034          TEMP2           = 0000003F
MC0             = 00000003          OP_SAVE_INDEX   = 0000002C          TEMP3           = 00000002
MC1             = 00000009          OP_SKIP         = 0000001C          TEMP4           = 00000002 R     F6
```

```
TEST            = 00000004
TEST_FLAG       = 00000000
TEST_HEAD       = 0000002D R     FD
TEST_KEY        = 00000000
TEST_LENGTH     = 00000003
TEST_SECT_HEAD  = 00000000 R     FD
THIS_TEST       = 00000062
TICK_FLAG       = 00000002
TICK_KEY        = 00000022
TOK             = 00000006
TRACE_ENABL     = 00000004
TRAP_HALT_EN    = 00000020
TRAP_OFF        = 00000002
TR_FLAG         = 00000080
TR_KEY          = 00000016
TSTSPAN_FLAG    = 00000040
UBI             = 00000006
UDP             = 0000001C
UTR             = 0000000D
VA_KEY          = 00000032
VBUS_CLOCK      = 00000080
VBUS_COMPARE    = 00000080
VBUS_LOAD       = 00000010
VBUS_SERIAL_IN  = 00000040
VBUS_SERIAL_OUT = 00000001
VB_KEY          = 00000026
V_BUS_STROBE    = 00000001
W               = 00000002
WBUS_FROM_D     = 00000020
WDREG           = 0000742C
WD_KEY          = 0000003A
WD_REG_BYTE_0   = 0000742C
WD_REG_BYTE_1   = 0000742D
WD_REG_BYTE_2   = 0000742E
WD_REG_BYTE_3   = 0000742F
WHREG           = 00007430
WH_REG_BYTE_0   = 00007430
WH_REG_BYTE_1   = 00007431
WH_REG_BYTE_2   = 00007432
WH_REG_BYTE_3   = 00007433
WORD            = 00000002
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
X_0055_D	00000002 (2.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0055_T	00000100 (256.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0055_GDCS	00000038 (56.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0055_ERTEMP	00000001 (1.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0055_FCACHE	00000001 (1.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0055_HFILL	00000044 (68.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
DIRECTORY	0000002C (44.)	07 (7.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0056_D	00000002 (2.)	08 (8.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0056_T	00000100 (256.)	09 (9.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0056_GDCS	00000038 (56.)	0A (10.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0056_ERTEMP	00000001 (1.)	0B (11.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0056_FCACHE	00000001 (1.)	0C (12.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0056_HFILL	00000044 (68.)	0D (13.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0057_D	00000002 (2.)	0E (14.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0057_T	00000100 (256.)	0F (15.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0057_GDCS	0000002D (45.)	10 (16.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0057_ERTEMP	00000001 (1.)	11 (17.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0057_FCACHE	00000001 (1.)	12 (18.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0057_HFILL	0000004F (79.)	13 (19.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0058_D	00000002 (2.)	14 (20.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0058_T	00000100 (256.)	15 (21.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0058_GDCS	0000002D (45.)	16 (22.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0058_ERTEMP	00000001 (1.)	17 (23.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0058_FCACHE	00000001 (1.)	18 (24.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0058_HFILL	0000004F (79.)	19 (25.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0059_D	00000002 (2.)	1A (26.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0059_T	00000100 (256.)	1B (27.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0059_GDCS	00000022 (34.)	1C (28.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0059_ERTEMP	00000005 (5.)	1D (29.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0059_FCACHE	00000001 (1.)	1E (30.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0059_HFILL	00000056 (86.)	1F (31.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0060_D	00000002 (2.)	20 (32.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0060_T	00000100 (256.)	21 (33.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0060_GDCS	00000022 (34.)	22 (34.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0060_ERTEMP	00000005 (5.)	23 (35.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0060_FCACHE	00000001 (1.)	24 (36.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0060_HFILL	00000056 (86.)	25 (37.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0061_D	00000002 (2.)	26 (38.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0061_T	00000100 (256.)	27 (39.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0061_GDCS	00000022 (34.)	28 (40.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0061_ERTEMP	00000005 (5.)	29 (41.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0061_FCACHE	00000001 (1.)	2A (42.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0061_HFILL	00000056 (86.)	2B (43.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0062_D	00000002 (2.)	2C (44.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0062_T	00000180 (384.)	2D (45.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0062_ERTEMP	00000015 (21.)	2E (46.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0062_FCACHE	00000001 (1.)	2F (47.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

X_0062_GDCS	00000001	(1.)	30	(48.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0062_HFILL	00000067	(103.)	31	(49.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0063_D	00000002	(2.)	32	(50.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0063_T	00000280	(640.)	33	(51.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0063_ERTEMP	00000011	(17.)	34	(52.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0063_FCACHE	00000001	(1.)	35	(53.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0063_GDCS	00000001	(1.)	36	(54.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0063_HFILL	0000006B	(107.)	37	(55.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0064_D	00000002	(2.)	38	(56.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0064_T	00000280	(640.)	39	(57.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0064_ERTEMP	00000011	(17.)	3A	(58.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0064_FCACHE	00000001	(1.)	3B	(59.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0064_GDCS	00000001	(1.)	3C	(60.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0064_HFILL	0000006B	(107.)	3D	(61.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0065_D	00000002	(2.)	3E	(62.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0065_T	00000280	(640.)	3F	(63.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0065_ERTEMP	00000011	(17.)	40	(64.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0065_FCACHE	00000001	(1.)	41	(65.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0065_GDCS	00000001	(1.)	42	(66.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0065_HFILL	0000006B	(107.)	43	(67.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0066_D	00000002	(2.)	44	(68.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0066_T	00000100	(256.)	45	(69.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0066_GDCS	0000002D	(45.)	46	(70.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0066_ERTEMP	00000005	(5.)	47	(71.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0066_FCACHE	00000001	(1.)	48	(72.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0066_HFILL	0000004B	(75.)	49	(73.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0067_D	00000002	(2.)	4A	(74.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0067_T	00000100	(256.)	4B	(75.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0067_GDCS	0000002D	(45.)	4C	(76.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0067_ERTEMP	00000005	(5.)	4D	(77.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0067_FCACHE	00000001	(1.)	4E	(78.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0067_HFILL	0000004B	(75.)	4F	(79.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0068_D	00000002	(2.)	50	(80.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0068_T	00000100	(256.)	51	(81.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0068_GDCS	0000002D	(45.)	52	(82.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0068_ERTEMP	00000005	(5.)	53	(83.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0068_FCACHE	00000001	(1.)	54	(84.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0068_HFILL	0000004B	(75.)	55	(85.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0069_D	00000002	(2.)	56	(86.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0069_T	00000100	(256.)	57	(87.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0069_GDCS	0000002D	(45.)	58	(88.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0069_ERTEMP	00000005	(5.)	59	(89.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0069_FCACHE	00000001	(1.)	5A	(90.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0069_HFILL	0000004B	(75.)	5B	(91.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0070_D	00000002	(2.)	5C	(92.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0070_T	00000180	(384.)	5D	(93.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0070_GDCS	0000002D	(45.)	5E	(94.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0070_ERTEMP	00000005	(5.)	5F	(95.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0070_FCACHE	00000001	(1.)	60	(96.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0070_HFILL	0000004B	(75.)	61	(97.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0071_D	00000002	(2.)	62	(98.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0071_T	00000100	(256.)	63	(99.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0071_GDCS	0000002D	(45.)	64	(100.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0071_ERTEMP	00000001	(1.)	65	(101.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0071_FCACHE	00000001	(1.)	66	(102.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE

X_0071_HFILL	0000004F	(79.)	67 (103.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0072_D	00000002	(2.)	68 (104.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0072_T	00000200	(512.)	69 (105.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0072_GDCS	00000022	(34.)	6A (106.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0072_ERTEMP	00000001	(1.)	6B (107.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0072_FCACHE	00000001	(1.)	6C (108.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0072_HFILL	0000005A	(90.)	6D (109.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0073_D	00000002	(2.)	6E (110.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0073_T	00000280	(640.)	6F (111.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0073_GDCS	00000038	(56.)	70 (112.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0073_ERTEMP	00000005	(5.)	71 (113.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0073_FCACHE	00000001	(1.)	72 (114.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0073_HFILL	00000040	(64.)	73 (115.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0074_D	00000002	(2.)	74 (116.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0074_T	00000300	(768.)	75 (117.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0074_GDCS	00000043	(67.)	76 (118.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0074_ERTEMP	00000005	(5.)	77 (119.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0074_FCACHE	00000001	(1.)	78 (120.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0074_HFILL	00000035	(53.)	79 (121.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0075_D	00000002	(2.)	7A (122.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0075_T	00000280	(640.)	7B (123.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0075_GDCS	00000038	(56.)	7C (124.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0075_ERTEMP	00000005	(5.)	7D (125.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0075_FCACHE	00000001	(1.)	7E (126.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0075_HFILL	00000040	(64.)	7F (127.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0076_D	00000002	(2.)	80 (128.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0076_T	00000300	(768.)	81 (129.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0076_GDCS	00000043	(67.)	82 (130.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0076_ERTEMP	00000005	(5.)	83 (131.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0076_FCACHE	00000001	(1.)	84 (132.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0076_HFILL	00000035	(53.)	85 (133.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0077_D	00000002	(2.)	86 (134.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0077_T	00000280	(640.)	87 (135.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0077_GDCS	00000038	(56.)	88 (136.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0077_ERTEMP	00000005	(5.)	89 (137.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0077_FCACHE	00000001	(1.)	8A (138.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0077_HFILL	00000040	(64.)	8B (139.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0078_D	00000002	(2.)	8C (140.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0078_T	00000300	(768.)	8D (141.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0078_GDCS	00000038	(56.)	8E (142.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0078_ERTEMP	00000005	(5.)	8F (143.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0078_FCACHE	00000001	(1.)	90 (144.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0078_HFILL	00000040	(64.)	91 (145.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0079_D	00000002	(2.)	92 (146.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0079_T	00000480	(1152.)	93 (147.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0079_GDCS	00000038	(56.)	94 (148.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0079_ERTEMP	0000000D	(13.)	95 (149.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0079_FCACHE	00000001	(1.)	96 (150.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0079_HFILL	00000038	(56.)	97 (151.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0080_D	00000002	(2.)	98 (152.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0080_T	00000300	(768.)	99 (153.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0080_GDCS	00000043	(67.)	9A (154.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0080_ERTEMP	0000000D	(13.)	9B (155.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0080_FCACHE	00000001	(1.)	9C (156.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0080_HFILL	0000002D	(45.)	9D (157.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE

X_0081_D	00000002	(2.)	9E	(158.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0081_T	00000200	(512.)	9F	(159.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0081_GDCS	0000002D	(45.)	A0	(160.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0081_ERTEMP	0000000D	(13.)	A1	(161.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0081_FCACHE	00000001	(1.)	A2	(162.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0081_HFILL	00000043	(67.)	A3	(163.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0082_D	00000002	(2.)	A4	(164.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0082_T	00000300	(768.)	A5	(165.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0082_GDCS	00000038	(56.)	A6	(166.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0082_ERTEMP	0000000D	(13.)	A7	(167.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0082_FCACHE	00000001	(1.)	A8	(168.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0082_HFILL	00000038	(56.)	A9	(169.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0083_D	00000002	(2.)	AA	(170.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0083_T	00000380	(896.)	AB	(171.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0083_GDCS	00000043	(67.)	AC	(172.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0083_ERTEMP	0000000D	(13.)	AD	(173.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0083_FCACHE	00000001	(1.)	AE	(174.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0083_HFILL	0000002D	(45.)	AF	(175.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0084_D	00000002	(2.)	B0	(176.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0084_T	00000180	(384.)	B1	(177.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0084_GDCS	00000022	(34.)	B2	(178.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0084_ERTEMP	00000001	(1.)	B3	(179.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0084_FCACHE	00000001	(1.)	B4	(180.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0084_HFILL	0000005A	(90.)	B5	(181.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0085_D	00000002	(2.)	B6	(182.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0085_T	00000180	(384.)	B7	(183.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0085_GDCS	00000022	(34.)	B8	(184.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0085_ERTEMP	00000001	(1.)	B9	(185.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0085_FCACHE	00000001	(1.)	BA	(186.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0085_HFILL	0000005A	(90.)	BB	(187.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0086_D	00000002	(2.)	BC	(188.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0086_T	00000280	(640.)	BD	(189.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0086_GDCS	00000043	(67.)	BE	(190.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0086_ERTEMP	0000000D	(13.)	BF	(191.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0086_FCACHE	00000001	(1.)	C0	(192.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0086_HFILL	0000002D	(45.)	C1	(193.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0087_D	00000002	(2.)	C2	(194.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0087_T	00000180	(384.)	C3	(195.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0087_GDCS	00000038	(56.)	C4	(196.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0087_ERTEMP	0000000D	(13.)	C5	(197.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0087_FCACHE	00000001	(1.)	C6	(198.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0087_HFILL	00000038	(56.)	C7	(199.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0088_D	00000002	(2.)	C8	(200.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0088_T	00000080	(128.)	C9	(201.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0088_GDCS	00000022	(34.)	CA	(202.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0088_ERTEMP	00000005	(5.)	CB	(203.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0088_FCACHE	00000001	(1.)	CC	(204.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0088_HFILL	00000056	(86.)	CD	(205.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0089_D	00000002	(2.)	CE	(206.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0089_T	00000180	(384.)	CF	(207.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0089_GDCS	00000038	(56.)	D0	(208.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0089_ERTEMP	00000009	(9.)	D1	(209.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0089_FCACHE	00000001	(1.)	D2	(210.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0089_HFILL	0000003C	(60.)	D3	(211.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0090_D	00000002	(2.)	D4	(212.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE

X_0090_T	00000180	(384.)	D5 (213.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0090_GDCS	00000038	(56.)	D6 (214.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0090_ERTEMP	00000009	(9.)	D7 (215.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0090_FCACHE	00000001	(1.)	D8 (216.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0090_HFILL	0000003C	(60.)	D9 (217.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0091_D	00000002	(2.)	DA (218.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0091_T	00000180	(384.)	DB (219.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0091_GDCS	0000002D	(45.)	DC (220.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0091_ERTEMP	00000005	(5.)	DD (221.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0091_FCACHE	00000001	(1.)	DE (222.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0091_HFILL	0000004B	(75.)	DF (223.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0092_D	00000002	(2.)	E0 (224.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0092_T	00000100	(256.)	E1 (225.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0092_GDCS	0000002D	(45.)	E2 (226.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0092_ERTEMP	00000005	(5.)	E3 (227.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0092_FCACHE	00000001	(1.)	E4 (228.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0092_HFILL	0000004B	(75.)	E5 (229.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0093_D	00000002	(2.)	E6 (230.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0093_T	0000017E	(382.)	E7 (231.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0094_D	00000002	(2.)	E8 (232.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0094_T	0000017E	(382.)	E9 (233.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0095_D	00000002	(2.)	EA (234.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0095_T	00000100	(256.)	EB (235.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0095_GDCS	0000002D	(45.)	EC (236.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0095_ERTEMP	00000001	(1.)	ED (237.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0095_FCACHE	00000001	(1.)	EE (238.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0095_HFILL	0000004F	(79.)	EF (239.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0096_D	00000002	(2.)	F0 (240.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0096_T	00000180	(384.)	F1 (241.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0096_GDCS	00000038	(56.)	F2 (242.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0096_ERTEMP	00000001	(1.)	F3 (243.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0096_FCACHE	00000001	(1.)	F4 (244.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0096_HFILL	00000044	(68.)	F5 (245.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0097_D	00000002	(2.)	F6 (246.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0097_T	00000100	(256.)	F7 (247.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0097_GDCS	0000002D	(45.)	F8 (248.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0097_ERTEMP	00000001	(1.)	F9 (249.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0097_FCACHE	00000001	(1.)	FA (250.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0097_HFILL	0000004F	(79.)	FB (251.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0098_D	00000002	(2.)	FC (252.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0098_T	0000017E	(382.)	FD (253.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0099_D	00000000	(0.)	FE (254.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	34	00:00:00.15	00:00:02.13
Command processing	856	00:00:00.92	00:00:02.61
Pass 1	5403	00:02:59.59	00:04:10.11
Symbol table sort	0	00:00:00.66	00:00:00.66
Pass 2	539	00:00:47.16	00:01:11.61
Symbol table output	2	00:00:00.25	00:00:01.29

Psect synopsis output	2	00:00:00.88	00:00:00.92
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	6837	00:03:49.62	00:05:29.33

The working set limit was 8000 pages.
2335716 bytes (4562 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 371 non-local and 243 local symbols.
9027 source lines were read in Pass 1, producing 555 object records in Pass 2.
97 pages of virtual memory were used to define 43 macros.

Macro library statistics

Macro library name	Macros defined
DISK$UCODPACK00:[VAX750]COMETMAC.MLB;805	43
SYS$COMMON:[SYSLIB]STARLET.MLB;2	0
TOTALS (all libraries)	43

1901 GETS were required to define 43 macros.

There were 0 errors, 1 warning and 0 information messages, on lines:
 4807 (1)

MACRO UCOD$0:[VAX750.DPM]TITLE+DISK$UCODPACK00:[VAX750]COMETASM+UCOD$0:[VAX750.DPM]ECKAB2/LIS=ECKAB2/OBJ=ECKAB2

(1) 24 "DPM Revision History"
(1) 1 ""
(1) 3 "DIAGNOSTIC MICROCODE MACROS"
(1) 98 " RDM Control File Macros"
(1) 109 " Diagnostic Macros"
(1) 194 ""
(1) 195 "MODULE GATE ARRAY MAPS"
(1) 197 " L0001 MULTIPLIER AND GATE ARRAY LOCATION"
(1) 254 " L0002 GATE ARRAY LOCATION"
(1) 311 " L0003 GATE ARRAY LOCATION"
(1) 369 " L0004 GATE ARRAY LOCATION"
(1) 427 " L0011 & L0016 GATE ARRAY LOCATION"
(1) 485 ""
(1) 490 "EQUATED SYMBOLS
(1) 493 ""
(1) 2 "TEST 063 Rotator Test 49 Convert BCD to Numeric Test
(1) 199 "TEST 064 Control store address test
(1) 399 "TEST 065 Micro Sequencer Stack Test
(1) 643 "TEST 066 D CLOCK INHIBIT ON BUT/RET.DINH
(1) 757 "TEST 067 MSQ adder test
(1) 925 "TEST 068 Check PCS RAM Data Integrity
(1) 1578 "TEST 069 Control store parity error micro-trap test
(1) 1753 "TEST 06A TEST XB0 AND XB1 REGISTERS
(1) 1999 "TEST 06B Prefetch Incrementer Test
(1) 2508 "TEST 06C Test Execution Buffer Rotate Logic
(1) 2822 "TEST 06D TEST CONSOLE CRAR REGISTER
(1) 2925 "TEST 06E TEST CONSOLE STATUS REGISTERS
(1) 3080 "TEST 06F TEST TU58 TRAR REGISTER
(1) 3182 "TEST 070 TEST TU58 STATUS REGISTERS
(1) 3315 "TEST 071 TEST IR AND OSR REGISTERS
(1) 3613 "TEST 072 TEST INSTRUCTION FETCH SIGNAL
(1) 3707 "TEST 073 TEST TRACE PENDING INTERRUPT
(1) 3834 "TEST 074 TEST CONSOLE HALT INTERRUPT
(1) 3957 "TEST 075 TEST NEXT INTERVAL REGISTER
(1) 4056 "TEST 076 TEST INTERVAL COUNT REGISTER
(1) 4163 "TEST 077 TIMER SERVICE INTERRUPT TEST 1
(1) 4350 "TEST 078 TIMER SERVICE INTERRUPT TEST 2
(1) 4493 "TEST 079 TIMER SERVICE INTERRUPT TEST 3
(1) 4637 "TEST 07A INTERVAL TIMER INTERRUPT TEST
(1) 4905 "TEST 07B TEST ARITHMETIC TRAP INTERRUPT
(1) 5070 "TEST 07C TEST POWER FAIL INTERRUPT
(1) 5204 "TEST 07D REI CHECK TEST 1
(1) 5447 "TEST 07E REI CHECK TEST 2
(1) 5676 "TEST 07F REI CHECK TEST 3
(1) 5818 "TEST 080 REI CHECK TEST 4
(1) 5958 "TEST 081 REI CHECK TEST 5
(1) 6113 "TEST 082 PC Increment on IRD1
(1) 6251 "TEST 083 NATIVE MODE DSIZE ROM TEST 1
(1) 6446 "TEST 084 NATIVE MODE DSIZE ROM TEST 2
(1) 7121 "TEST 085 NATIVE MODE DSIZE ROM TEST 3
(1) 7795 "TEST 086 NATIVE MODE DSIZE ROM TEST 4
(1) 8469 "TEST 087 NATIVE MODE DSIZE ROM TEST 5
(1) 9143 "TEST 088 NATIVE MODE DSIZE ROM TEST 6
(1) 9817 "TEST 089 NATIVE MODE DSIZE ROM TEST 7
(1) 10491 "TEST 08A NATIVE MODE DSIZE ROM TEST 8
(1) 11165 "TEST 08B NATIVE MODE DSIZE ROM TEST 9
(1) 11838 "TEST 08C NATIVE MODE DSIZE ROM TEST 10

```
    (1)     12510           "TEST   08D     NATIVE MODE DSIZE ROM TEST 11
    (1)     13183           "TEST   08E     NATIVE MODE DSIZE ROM TEST 12
```

```
0000     1          .TITLE  ECKAB VAX-11/750 DPM MICRO DIAGNOSTICS
0000     2          .IDENT  /V09.01/
0000     3 ;
0000     4 ; COPYRIGHT (C) 1980,1983
0000     5 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     6 ;
0000     7 ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE  ONLY ON A SINGLE
0000     8 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000     9 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    10 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    11 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    12 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    13 ; REMAIN IN DEC.
0000    14 ;
0000    15 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    16 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    17 ; CORPORATION.
0000    18 ;
0000    19 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    20 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    21 ;
0000    22 ;
```

```
0000     24          .SBTTL  "DPM Revision History"
0000     25 ;++
0000     26 ; DPM MICRO-DIAGNOSTIC PACKAGE
0000     27 ;
0000     28 ; REVISION HISTORY
0000     29 ;
0000     30 ; 00-01 Tom Groetzinger 4-FEB-1980
0000     31 ;       Started file.
0000     32 ;
0000     33 ; 00-02 Dave Spain      27-FEB-1980
0000     34 ;       Added CS BUS INTEGRITY test.
0000     35 ;
0000     36 ; 00-03 Tom Groetzinger 12-MAR-1980
0000     37 ;       Added five tests.
0000     38 ;
0000     39 ; 00-04 Tom Groetzinger 17-MAR-1980
0000     40 ;       Added SRK,SRM callouts to some tests.
0000     41 ;
0000     42 ; 00-05 Tom Groetzinger 17-MAR-1980
0000     43 ;       Improved tests 7 and 16 to increase coverage due to Westboro feedback
0000     44 ;
0000     45 ; 00-06 Tom Groetzinger 17-MAR-1980
0000     46 ;       Debugged test 16 dueu to major rewrite
0000     47 ;
0000     48 ; 00-07 Tom Groetzinger 17-MAR-1980
0000     49 ;       Added 9 Super Rotator tests
0000     50 ;
0000     51 ; 01-08 Dave Spain       4-APR-80
0000     52 ;       Added 4 Micro-sequencer tests for offical release.
0000     53 ;
0000     54 ; 01-09 Bill Landry     4-APR-1980
0000     55 ;       Added the XB, PREFETCH and IRD tests from ECKAC.SRC so
0000     56 ;       manufacturing can run one tape for DPM.
0000     57 ;
0000     58 ; 00-10 Bill Landry     4-APR-1980
0000     59 ;       Let's save version 1.0 for our first SDC release.
0000     60 ;
0000     61 ; 00-11 Tom Groetzinger 7-APR-1980
0000     62 ;       Added 2 ALK Tests
0000     63 ;
0000     64 ; 00-12 Tom Groetzinger 8-APR-1980
0000     65 ;       Deleted last 2 D Register outputs from ALK Test #6
0000     66 ;
0000     67 ; 00-13 Tom Groetzinger 14-APR-1980
0000     68 ;       Added D Register output to ALK test #6
0000     69 ;
0000     70 ; 00-14 Dave Spain      25-APR-1980
0000     71 ;       Changed mask on the micro-sequencer tests to ignore bit 13 of the
0000     72 ;       Control Store address.
0000     73 ;
0000     74 ; 00-15 Dave Spain      28-APR-1980
0000     75 ;       Fixed bug in ALK Tests 6 and 7 by replacing VSIZE/1 micro order with a
0000     76 ;       SIZE[LONG] macro in micro words 6 and 3 respectively.
0000     77 ;
0000     78 ; 00-16 Dave Spain      29-APR-1980
```

```
0000     79 ;       Changed mask on CS address for Instruction Address Decode test from
0000     80 ;       3FFF to 1FFF to ignore bit 13 of Control Store address.
0000     81 ;
0000     82 ; 00-17 Dave Spain      30-APR-1980
0000     83 ;       Added CCS module to error callout for Control Store Bus Integrity test.
0000     84 ;
0000     85 ; 00-18 Dave Spain       9-MAY-1980
0000     86 ;       Added SRM gate array to error callout routine in RTMP0 and RTEMP
0000     87 ;       Sratchpad tests.
0000     88 ;
0000     89 ; 00-19 Bill Landry     10-JUN-1980
0000     90 ;       Added RDM tests to beginning of tape
0000     91 ;
0000     92 ; 01-00 Bill Landry/Dave Spain  13-JUN-1980
0000     93 ;       Added macro file to title section and compatibility mode IRD tests
0000     94 ;       then submitted to RELEASE ENGINERRING.
0000     95 ;
0000     96 ; 01-01 Dave Spain      19-JUN-1980
0000     97 ;       Fixed prefetch incrementer test to clear out cache data after every
0000     98 ;       prefetch. This will prevent the test from mistaking bad addresses from
0000     99 ;       good ones.
0000    100 ;
0000    101 ; 01-02 Dave Spain       2-JUL-1980
0000    102 ;       Added CM Second IRDX tests to file.
0000    103 ;
0000    104 ; 01-03 Dave Spain      18-AUG-1980
0000    105 ;       Added CM branch test to file.
0000    106 ;
0000    107 ; 01-04 Dave Spain      10-SEPT-1980
0000    108 ;       Fixed "DEFAULT MTEMP0 TEST"
0000    109 ;
0000    110 ; 01-05 Bill Landry     11-SEPT-1980
0000    111 ;       Revised RDM to CMI tests
0000    112 ;
0000    113 ; 02-00 Dave Spain      28-SEPT-1980
0000    114 ;       Added rotator tests to file and submitted to SDC.
0000    115 ;
0000    116 ; 02-01 Dave Spain      28-OCT-1980
0000    117 ;       Added QV flag to tests to be run by manufacturing.
0000    118 ;
0000    119 ; 02-02 Bill Landry     10-NOV-1980
0000    120 ;       Added tests for TOK and SAC gate arrays.
0000    121 ;
0000    122 ; 02-03 Bill Landry     13-NOV-1980
0000    123 ;       Added test of REI CHECK
0000    124 ;
0000    125 ; 03-00 Dave Spain      17-NOV-1980
0000    126 ;       Added rotator tests 23-28 and released to SDC.
0000    127 ;
0000    128 ; 03-01 Dave Spain      10-DEC-1980
0000    129 ;       Added rotator tests 29-42.
0000    130 ;
0000    131 ; 03-02 Bill Landry     11-DEC-1980
0000    132 ;       Fixed bug in CS ADDRESS test
0000    133 ;
```

```
0000    134 ; 03-03 Dave Spain      12-DEC-1980
0000    135 ;       Enhanced Rotator Test 1 to call out ALP slices.
0000    136 ;
0000    137 ; 03-04 Bill Landry     16-DEC-1980
0000    138 ;       Added DSIZE ROM tests
0000    139 ;
0000    140 ; 03-05 Bill Landry     31-DEC-1980
0000    141 ;       Last great act of 80. Separated DPM and RDM tests into two files.
0000    142 ;
0000    143 ; 04-00 Bill Landry     5-JAN-1981
0000    144 ;       Submitted to RELEASE ENGINERRING
0000    145 ;
0000    146 ; 04-01 Bill Landry     14-JAN-1981
0000    147 ;       Revised INTERVAL TIMER INTERRUPT TEST to fix bug caused by INT PEND
0000    148 ;       ECO
0000    149 ;
0000    150 ; 04-02 Dave Spain      16-JAN-1981
0000    151 ;       Fixed error callout in CS Parity Error, XB, and Prefetch Incrementer
0000    152 ;       tests and added seven new rotator tests.
0000    153 ;
0000    154 ; 04-03 Bill Landry     4-FEB-1981
0000    155 ;       Added tests for CCC gate array
0000    156 ;
0000    157 ; 04-04 Dave Spain      9-FEB-1981
0000    158 ;       Changed the names of rotator tests 23-33.
0000    159 ;
0000    160 ; 04-05 Bill Landry     24-FEB-1981
0000    161 ;       Added tests for DPM discrete branching logic.
0000    162 ;
0000    163 ; 04-06 Bill Landry     27-FEB-1981
0000    164 ;       Added tests for branching logic in PHB gate array.
0000    165 ;
0000    166 ; 05-00 Dave Spain       1-MAR-1981
0000    167 ;       Added 5 SRK status rotator tests and submitted to SDC.
0000    168 ;
0000    169 ; 05-01 Dave Spain      25-MAR-1981
0000    170 ;       Fixed a loop on test bug in the Alu Functions Test by adding a LDFIELD
0000    171 ;       pseudo-op that reinitializes the ROT field in Subtest 1.
0000    172 ;
0000    173 ; 05-02 Bill Landry     25-MAR-1981
0000    174 ;       Fixed bug in BUT/NO.FPA test.  I was branching on the wrong conditions.
0000    175 ;
0000    176 ; 05-03 Dave Spain      14-APR-1981
0000    177 ;       Cleaned up documentation for R Scratchpads Write All Cells Test.
0000    178 ;
0000    179 ; 06-00 Bill Landry     2-MAY-1981
0000    180 ;       Added SPA branching, CON registers, TU58 registers, and CCS PARITY
0000    181 ;       tests.  Submitted to Release Engineering.
0000    182 ;
0000    183 ; 06-01 Dave Spain      17-JUN-1981
0000    184 ;       Replaced the old IRD1 Instruction Address Decode test with two new
0000    185 ;       tests that toggle the PSL's FPD bit and can handle an FPA.
0000    186 ;
0000    187 ; 06-02 Bill Landry     19-AUG-1981
0000    188 ;       Fixed bugs created by introduction of FPA option.
```

```
0000   189 ;       Replaced all references to COMET with the word CPU.
0000   190 ;       Fixed typing error in vbus data of CPU CLOCK TEST.
0000   191 ;       Submitted to RELEASE ENGINEERING.
0000   192 ;
0000   193 ; 06-03 Bill Landry     24-SEPT-1981
0000   194 ;       Fixed potential clock problem with FPA in Z BIT TEST 1.
0000   195 ;
0000   196 ; 06-04 Bill Landry     9-OCT-1981
0000   197 ;       Added changes for ECO L0002-TW002
0000   198 ;
0000   199 ; 07-00 Bill Landry     31-OCT-1981
0000   200 ;       Removed changes for ECO L0002-TW002.  ECO is on hold.
0000   201 ;       Added tests for FPA to CPU interface logic
0000   202 ;       Moved WCS test to MIC tape
0000   203 ;       Submitted to RELEASE ENGINEERING.
0000   204 ;
0000   205 ; 07-01 Bill Landry     13-NOV-1981
0000   206 ;       Stopped release of 7.0
0000   207 ;       Added FPA TRAP LOGIC TEST
0000   208 ;       Enhanced error messages for FPA
0000   209 ;       Resubmitted to RELEASE ENGINEERING
0000   210 ;
0000   211 ; 07-02 Bill Landry     20-NOV-1981
0000   212 ;       Stopped release of 7.1
0000   213 ;       Enhanced FPA ENABLE/DISABLE TEST to check for MSEQ INIT
0000   214 ;       Resubmitted to RELEASE ENGINEERING
0000   215 ;
0000   216 ; 07-03 Dave Shull      26-Mar-1982
0000   217 ;       Added the ERRLOOP pseudo to the Rotator Extract tests which were
0000   218 ;       missing these and couldn't be looped on.
0000   219 ;
0000   220 ; 07-04 Dave Shull      05-Apr-1982
0000   221 ;       Removed all BEGINSA and ENDSA pseudo's.
0000   222 ;       Submitted to RELEASE ENGINEERING
0000   223 ;
0000   224 ; 07-05 Dave Shull      21-May-1982
0000   225 ;       Add a subtest to the Micro Sequencer Stack Test that will test bits
0000   226 ;       11 and 12 of the micro stack.
0000   227 ;
0000   228 ; 07-06 Dave Shull      24-May-1982
0000   229 ;       Changed documentation in the CM Odd Address BUT Test to reflect the
0000   230 ;       ECO L0002-TW002
0000   231 ;
0000   232 ; 08-00 Dave Shull      18-Jun-1982
0000   233 ;       Added 10 new tests.  Two of those tests are an addition to the existing
0000   234 ;       IRD1 rom tests.  These 2 new tests check the signal ROM OS INH H.
0000   235 ;       The other 8 new tests check the validity of the IRDX ROMs.  They check
0000   236 ;       both the address and the signal ROM OS INH H.
0000   237 ;
0000   238 ; 08-01 Dave Shull      22-Jun-1982
0000   239 ;       Added a subtest to the Register Backup Stack test that will check the
0000   240 ;       bit that records Increment/Decrement of the GPR.
0000   241 ;
0000   242 ; 09-00 Dave Shull      03-Nov-1982
0000   243 ;       Added two new Tests for coverage the new PCS (L0008) module.
```

```
0000   244 ;
0000   245 ; 09-01 Dave Smith      15-May-1983
0000   246 ;       Correct DCS in ALK test 3,6,7. Disabled ALP's on the WBUS.
0000   247 ;--
0000     1          .SBTTL  ""
```

```
0000          .SBTTL  "DIAGNOSTIC MICROCODE MACROS"
0000    ;++
0000    ; Revision History
0000    ;
0000    ; 28    Dave Spain      22-Jun-82
0000    ;       Removed LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
0000    ;       Decided not to use them.
0000    ;
0000    ; 27    Dave Spain      15-Jun-82
0000    ;       Added LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
0000    ;
0000    ; 26    Dave Spain      19-May-82
0000    ;       Added M[]_FPA macro.
0000    ;
0000    ; 25    Rich Lewis      23-Feb-82
0000    ;       Added PSL_WB, M[]_R[].OR.NIB0(M), WB_FPA CCR, WB_FPA TCR, Q_R[].ASL.P,
0000    ;       RDM_M[].XZ.OR.R[], FPA_WB macros
0000    ;
0000    ; 24    Dave Spain      02-Feb-82
0000    ;       Added FPA_R[]+M[] macro.
0000    ;
0000    ; 23    Bill Landry     22-OCT-81
0000    ;       Added PSL_RDM macro
0000    ;
0000    ; 22    Dave Spain       5 Oct 81
0000    ;       Added RDM_FPA and FPA_M[] FPA_RDM macros.
0000    ;
0000    ; 21    Bill Landry     29 Sept 81
0000    ;       Added FPA macro, FPA_M[] FPA_R[].
0000    ;
0000    ; 20    Dave Spain      14 Aug 81
0000    ;       Added the TESTFPA [] macro.
0000    ;
0000    ; 19    Dave Spain      18 July 81
0000    ;       Added the following FPA macros, FPA_RDM and FPA_R[].
0000    ;
0000    ; 18    Dave Spain      24 Sept 80
0000    ;       Added RDM_Q macro. This macro uses the ALU.
0000    ;
0000    ; 17    Bill Landry     23 Sept 80
0000    ;       Added macro VA_VA-ZLIT0[] for testing memory in decending order.
0000    ;
0000    ; 16    Dave Spain      27 August 80
0000    ;       Changed RDM_Q macro to RDM_Q Q_D. This more accurately reflects macro.
0000    ;
0000    ; 15    Bill Landry     4 June 80
0000    ;       Deleted BUS/PRB.RD.PTE from macro CLRTB.VA_VA and CLRCH.VA_VA to
0000    ;       satisfy validity checks.
0000    ;
0000    ; 14    Dave Spain      23 May 80
0000    ;       Added BUS/PRB.RD.PTE to the clear cache and clear TB macros for Mr.
0000    ;       Bill.
0000    ;
0000    ; 13    Dave Spain      14 Apr 80
0000    ;       Added SIZE[] or VSIZE/1 to some macros. This allows version 65 of
```

```
0000     58 ;       the COMET micro-code defines to run.
0000     59 ;
0000     60 ; 12    Bill Landry      1 Apr 80
0000     61 ;       Added BUS/PRB.RD micro order to RDM_TB macro to compensate for timing
0000     62 ;       problem in ADD gate array.  This is not an APRIL FOOL!
0000     63 ;
0000     64 ; 11    Dave Spain      27 Mar 80
0000     65 ;       Fixed TB macro's to include MSRC/VA to prevent validity failure.
0000     66 ;
0000     67 ; 10    Bill Landry     19 Mar 80
0000     68 ;       Added Q_R[].RL.P and VA_Q.OR.R[]
0000     69 ;
0000     70 ; 09    Bill Landry     19 Mar 80
0000     71 ;       Deleted TB_R[].RL.P.OR.D
0000     72 ;
0000     73 ; 08    Bill Landry     18 Mar 80
0000     74 ;       Added VA_R[].RL.P
0000     75 ;
0000     76 ; 07    Bill Landry     07 Mar 80
0000     77 ;       Added Q_M[].RL.PTE
0000     78 ;
0000     79 ; 06    Dave Spain      06 Mar 80
0000     80 ;       Changed TB_R[].RL.P macro to TB_R[].RL.P.OR.D.
0000     81 ;
0000     82 ; 05    Dave Spain      03 Mar 80
0000     83 ;       Added TB_R[].RL.P macro for Bill Landry.
0000     84 ;
0000     85 ; 04    Dave Spain      22 Feb 80
0000     86 ;       Added CLRTB.VA_R[] macro for Bill Landry.
0000     87 ;
0000     88 ; 03    Dave Spain      30 Jan 80
0000     89 ;       Added Tom Groetzinger's macro definitions.
0000     90 ;
0000     91 ; 02    Dave Spain      30 Jan 80
0000     92 ;       Added Tony Vezza's macro definitions.
0000     93 ;
0000     94 ; 01    Dave Spain      29 Jan 80
0000     95 ;       Macros initiated.
0000     96 ;--
0000     97
0000     98          .SBTTL  "       RDM Control File Macros"
0000     99
0000    100 ;STROBE.V.BUS            "RDMCF0/V.STROBE"
0000    101 ;DCS.ADDR_0              "RDMCF1/DCS.ADDR.CLR"
0000    102 ;STOP.CLOCK              "RDMCF2/STOP.CPU.CLOCK"
0000    103 ;LATCH.CS.ADDR           "RDMCF3/DIS.STROBE.CS.A"
0000    104 ;RDM_WB                  "RDMCF4/H.REG.FROM.WB"
0000    105 ;WB_RDM                  "RDMCF5/WB.FROM.D.REG"
0000    106 ;RDM_CMI                                "RDMCF6/STROBE.CMI"
0000    107 ;INH.CLOCK.SA            "RDMCF7/INH.SA.CLOCK"
0000    108
0000    109          .SBTTL  "       Diagnostic Macros"
0000    110
0000    111 ;CLOCK.EXT                              "CLKX/XTND"
0000    112 ;CLRCH.VA_RDM                           "WB_RDM,WCTRL/CLRCH.VA_WB,BUS/PRB.RD.PTE"
```

```
0000    113 ;CLRCH.VA_VA                    "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRCH.VA_WB
0000    114 ;CLRTB.VA_D                     "RSRC/ZERO,MUX/D.R1,ALU/OR,WCTRL/CLRTB.VA_WB,
0000    115 ;                                       BUS/PRB.RD.PTE"
0000    116 ;CLRTB.VA_RDM                   "WB_RDM,WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
0000    117 ;CLRTB.VA_R[]                   "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/CLRTB.VA_WB,
0000    118 ;                                       BUS/PRB.RD.PTE"
0000    119 ;CLRTB.VA_VA                    "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRTB.VA_WB
0000    120 ;CLRTB.VA_WB                    "WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
0000    121 ;D_LONLIT                       "RSRC/LONLIT,ALPCTL/WX_D_R.Q_D"
0000    122 ;D_M[]_LONLIT                   "D_LONLIT,MSRC/a1"
0000    123 ;D_VA_0                         "RSRC/ZERO,ROT/ZERO,MUX/R.S,ALU/OR,DQ1/D_WX,
0000    124 ;                                       WCTRL/VA_WB"
0000    125 ;FPA_M[] FPA_RDM                        "MSRC/a1,WB_RDM,FPA/FPA_MBUS.FPA_WBUS"
0000    126 ;FPA_M[] FPA_R[]                        "FPA/FPA_MBUS.FPA_WBUS,MSRC/a1,RSRC/a2,MUX/R
0000    127 ;                                       ROT/ZERO,ALU/OR"
0000    128 ;FPA_RDM                                "WB_RDM,FPA/FPA_DATA.WBUS"
0000    129 ;FPA_R[]                                "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,FPA/FPA_DAT
0000    130 ;FPA_R[]+M[]                    "RSRC/a1,MSRC/a2,MUX/M.R1,ALU/A+B+CI,ALUCI/ZERO,FPA/
0000    131 ;FPA_WB                         "FPA/FPA_DATA.WBUS"
0000    132 ;MEMSCAR_R[]                    "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/MEMSCAR_WB"
0000    133 ;M[]_FPA                                "MSRC/a1,SPW/MLONG,FPA/WBUS_FPA"
0000    134 ;M[]_LONLIT                     "MSRC/a1,SPW/MLONG,ALPCTL/WX_R.Q_D,RSRC/LONLIT"
0000    135 ;M[]_R[].OR.NIB0(M)             "MSRC/a1,RSRC/a2,SPW/MLONG,ROT/GETNIB,ALU/OR,
0000    136 ;                                       MUX/R.S,DQ1/NOP"
0000    137 ;M[]_RDM                                "WB_RDM,SPW/MLONG,MSRC/a1"
0000    138 ;PC_PC+RDM                      "WCTRL/PC_PC+WB,WB_RDM"
0000    139 ;PC_PC+4 MB_XB                  "MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,SIZE[LONG]"
0000    140 ;PC_PC+4 RDM_XB                 "RSRC/ZERO,MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,
0000    141 ;                                       MUX/M.R1,ALU/OR,RDM_WB,SIZE[LONG]"
0000    142 ;PC_RDM                         "WB_RDM,WCTRL/PC_WB"
0000    143 ;PSL_RDM                                "CCPSL/PSL_WB.CCBR_ALUS,WB_RDM"
0000    144 ;PSL_WB                         "CCPSL/PSL_WB.CCBR_ALUS"
0000    145 ;Q_D WX_R                       "ALPCTL/WX_R.Q_D"
0000    146 ;Q_LONLIT                       "RSRC/LONLIT,ALPCTL/WX_S.Q_R"
0000    147 ;Q_M[].RL.PTE                   "ALPCTL/WX_Q_S,MSRC/a1,ROT/RL.MM.PTE"
0000    148 ;Q_R[].AND.Q                    "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/AND"
0000    149 ;Q_R[].ASL.P                    "ALPCTL/WX_Q_S,RSRC/a1,ROT/ASL.R.P"
0000    150 ;Q_R[].OR.Q                     "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/OR"
0000    151 ;Q_R[].RL.P                     "ALPCTL/WX_Q_S,RSRC/a1,ROT/RL.RR.P"
0000    152 ;RDM_ALUF                       "ALPCTL/WB_ALUF,RDM_WB"
0000    153 ;RDM_FPA                                "FPA/WBUS_FPA,RDM_WB"
0000    154 ;RDM_LONLIT                     "WB_LONLIT,RDM_WB"
0000    155 ;RDM_LONLIT.Q_M                 "RSRC/LONLIT,ALPCTL/WX_R.Q_M,RDM_WB"
0000    156 ;RDM_M[]                                "MSRC/a1,ROT/ZERO,MUX/M.S,ALU/OR,RDM_WB"
0000    157 ;RDM_M[].OR.R[]                 "WB_M[a1].OR.R[a2],RDM_WB"
0000    158 ;RDM_M[]-R[]                    "WB_M[a1]-R[a2],RDM_WB"
0000    159 ;RDM_M[].XZ.OR.R[]              "ALU/A+B+CI,MUX/R.S,ROT/XZ.MM,MSRC/a1,RSRC/a2,
                                            RDM_WB"
0000    160 ;
0000    161 ;RDM_PC                         "MSRC/PC,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
0000    162 ;RDM_PCBACK                     "RSRC/ZERO,MSRC/PCBACK,MUX/M.R1,ALU/OR,RDM_WB"
0000    163 ;RDM_Q                          "RSRC/ZERO,MUX/R.Q,ALU/OR,RDM_WB"
0000    164 ;RDM_Q Q_D                      "ALPCTL/WX_Q.Q_D,RDM_WB"
0000    165 ;RDM_R[]                                "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,RDM_WB"
0000    166 ;RDM_TB                         "WB_M[TB],RDM_WB,BUS/PRB.RD,SIZE[LONG]"
0000    167 ;RDM_VA                         "MSRC/VA,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
```

```
0000   168 ;RNUM_LONLIT                    "ALPCTL/WX_D_R.Q_M,MSRC/RNUM_WBUS,RSRC/LONLIT"
0000   169 ;R[]_RDM                              "WB_RDM,SPW/RLONG,RSRC/a1"
0000   170 ;R[].DT_Q                      "RSRC/a1,D_Q Q_D,SPW/RSIZE,VSIZE/1"
0000   171 ;R[]_VA                       "RSRC/a1,MSRC/VA,ROT/ZERO,MUX/M.S,ALU/OR,SPW/RLONG"
0000   172 ;R[]_WB                       "RSRC/a1,SPW/RLONG"
0000   173 ;TB_RDM                       "WCTRL/TB_WB,WB_RDM,MSRC/VA"
0000   174 ;TB_WB                        "WCTRL/TB_WB,MSRC/VA"
0000   175 ;TESTFPA []                    "TESTFPA/a1"
0000   176 ;VA_PC+LONLIT+I.PC_PC+I       "RSRC/LONLIT,MSRC/XB.PC_PC+I,ROT/ZERO,MUX/R.S,ALU/OR
0000   177 ;                                      ISTRM/ISIZE_DSIZE,WCTRL/VA_PC+I+W.PC_PC+I,
0000   178 ;                                      SIZE[LONG]"
0000   179 ;VA_Q.OR.R[]                   "WCTRL/VA_WB,RSRC/a1,ALU/OR,MUX/R.Q"
0000   180 ;VA_RDM                       "WB_RDM,WCTRL/VA_WB"
0000   181 ;VA_R[].RL.P                  "ALPCTL/WX_S,ROT/RL.RR.P,RSRC/a1,WCTRL/VA_WB"
0000   182 ;VA_VA-ZLIT0[]                "WCTRL/VA_WB,LIT/LITRL,LITRL/a1,ROT/ZLIT0,MSRC/VA,
0000   183 ;                                      MUX/M.S,ALU/A-B-CI"
0000   184 ;VA_WB                        "WCTRL/VA_WB"
0000   185 ;WB_FPA CCR                   "FPA/WBUS_FPA.CC"
0000   186 ;WB_FPA TCR                   "WCTRL/FPTCR"
0000   187 ;WB_LONLIT                    "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
0000   188 ;WB_LONLIT.Q_M                "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
0000   189 ;WB_M[]+R[]                   "ALU/A+B+CI,ALUCI/ZERO,MUX/M.R1,MSRC/a1,RSRC/a2"
0000   190 ;WB_R[].Q_D                   "RSRC/a1,ALPCTL/WX_R.Q_D"
0000   191 ;WDR_R[]                             "WCTRL/WDR_WB,ALU/OR,MUX/R.S,RSRC/a1,ROT/ZER
0000   192 ;WRITE.LONG R[]               "BUS/WRITE.LNG,WCTRL/WDR_WB.UR,RSRC/a1,ROT/ZERO,
0000   193 ;                                      MUX/R.S,ALU/OR,SIZE[LONG]"
0000   194         .SBTTL  ""
0000   195         .SBTTL  "MODULE GATE ARRAY MAPS"
```

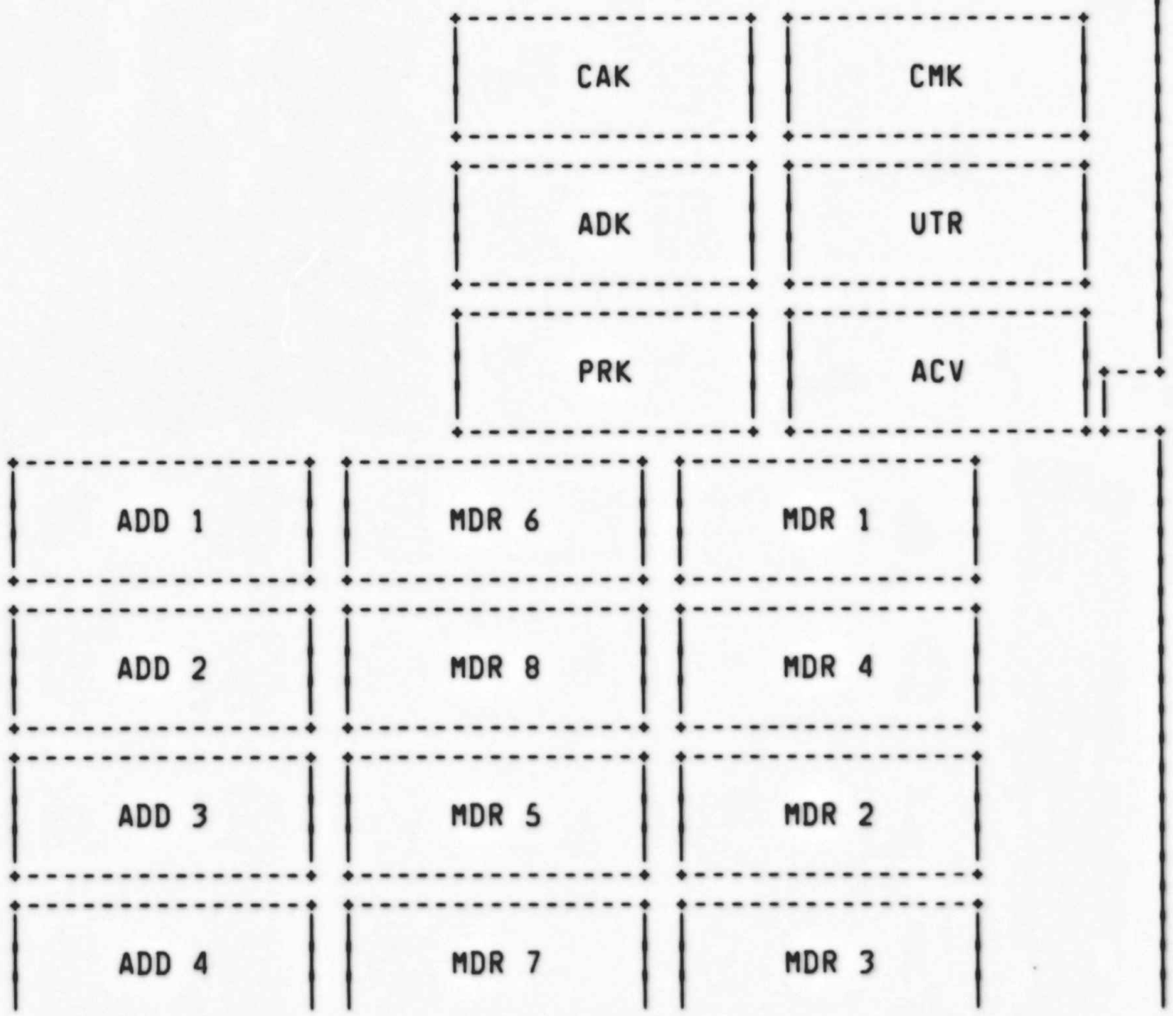
```
0000          .SBTTL  "      L0001 MULTIPLIER AND GATE ARRAY LOCATION"
0000  ;+-----------------------------------------------------------------------------+
0000  ;|  +---------------+ +---------------+ +---------------+ +---------------+   |
0000  ;|  |               | |               | |               | |               |   |
0000  ;|  |      FCC      | |      FQA      | |     FEX 0     | |     FEX 1     |   |
0000  ;|  |               | |               | |               | |               |   |
0000  ;|  +---------------+ +---------------+ +---------------+ +---------------+   |
0000  ;|                                                                             |
0000  ;|                                                                             |
0000  ;|                                                                             |
0000  ;|                                                                             |
0000  ;|                                                                             |
0000  ;|                     +---------------+ +---------------+ +---------------++--+
0000  ;|                     |               | |               | |               ||
0000  ;|                     |     FFA 5     | |     FFA 0     | |     FIO 1     |+--+
0000  ;|                     |               | |               | |               |   |
0000  ;|                     +---------------+ +---------------+ +---------------+   |
0000  ;|  +---------------+ +---------------+ +---------------+ +---------------+   |
0000  ;|  |    MULT 4     | |               | |               | |               |   |
0000  ;|  +---------------+ |     FFA 4     | |     FCS 1     | |     FIO 0     |   |
0000  ;|  +---------------+ |               | |               | |               |   |
0000  ;|  |    MULT 1     | +---------------+ +---------------+ +---------------+   |
0000  ;|  +---------------+ +---------------+ +---------------+ +---------------+   |
0000  ;|                     |               | |               | |               |   |
0000  ;|                     |    ALU CLA    | |     FCS 2     | |     FMR 1     |   |
0000  ;|                     |               | |               | |               |   |
0000  ;|                     +---------------+ +---------------+ +---------------+   |
0000  ;|  +---------------+ +---------------+ +---------------+ +---------------+   |
0000  ;|  |    MULT 5     | |               | |               | |               |   |
0000  ;|  +---------------+ |     FFA 1     | |     FCS 0     | |     FMR 0     |   |
0000  ;|  +---------------+ |               | |               | |               |   |
0000  ;|  |    MULT 2     | +---------------+ +---------------+ +---------------++--+
0000  ;|  +---------------+ +---------------+ +---------------+ +---------------+|
0000  ;|                     |               | |               | |               |+--+
0000  ;|                     |     FFA 2     | |     FCS 3     | |     FIO 2     |   |
0000  ;|                     |               | |               | |               |   |
0000  ;|                     +---------------+ +---------------+ +---------------+   |
0000  ;|  +---------------+ +---------------+ +---------------+ +---------------+   |
0000  ;|  |    MULT 6     | |               | |               | |               |   |
0000  ;|  +---------------+ |     FFA 6     | |     FIO 3     | |     FIO 5     |   |
0000  ;|  +---------------+ |               | |               | |               |   |
0000  ;|  |    MULT 3     | +---------------+ +---------------+ +---------------+   |
0000  ;|  +---------------+ +---------------+ +---------------+ +---------------+   |
0000  ;|                     |               | |               | |               |   |
0000  ;|                     |     FFA 3     | |    INC CLA    | |     FIO 4     |   |
0000  ;|                     |               | |               | |               |   |
0000  ;|                     +---------------+ +---------------+ +---------------+   |
0000  ;|  +---------------+ +---------------+ +---------------+ +---------------+   |
0000  ;|  |    MULT 7     | |               | |               | |               |   |
0000  ;|  +---------------+ |     FFA 7     | |     FIO 7     | |     FIO 6     |   |
0000  ;|                     |               | |               | |               |   |
0000  ;|                     +---------------+ +---------------+ +---------------+   |
0000  ;|                                                                             |
0000  ;|                                                                             |
0000  ;|                                                                             |
```

```
                    0000   252 ;+-----------------------------------------------------------------------------+
```

```
                        .SBTTL  "       L0002 GATE ARRAY LOCATION"
;+-----------------------------------------------------------------------------+
;|                                                                             |
;|                                                                             |
;|                                                                             |
```

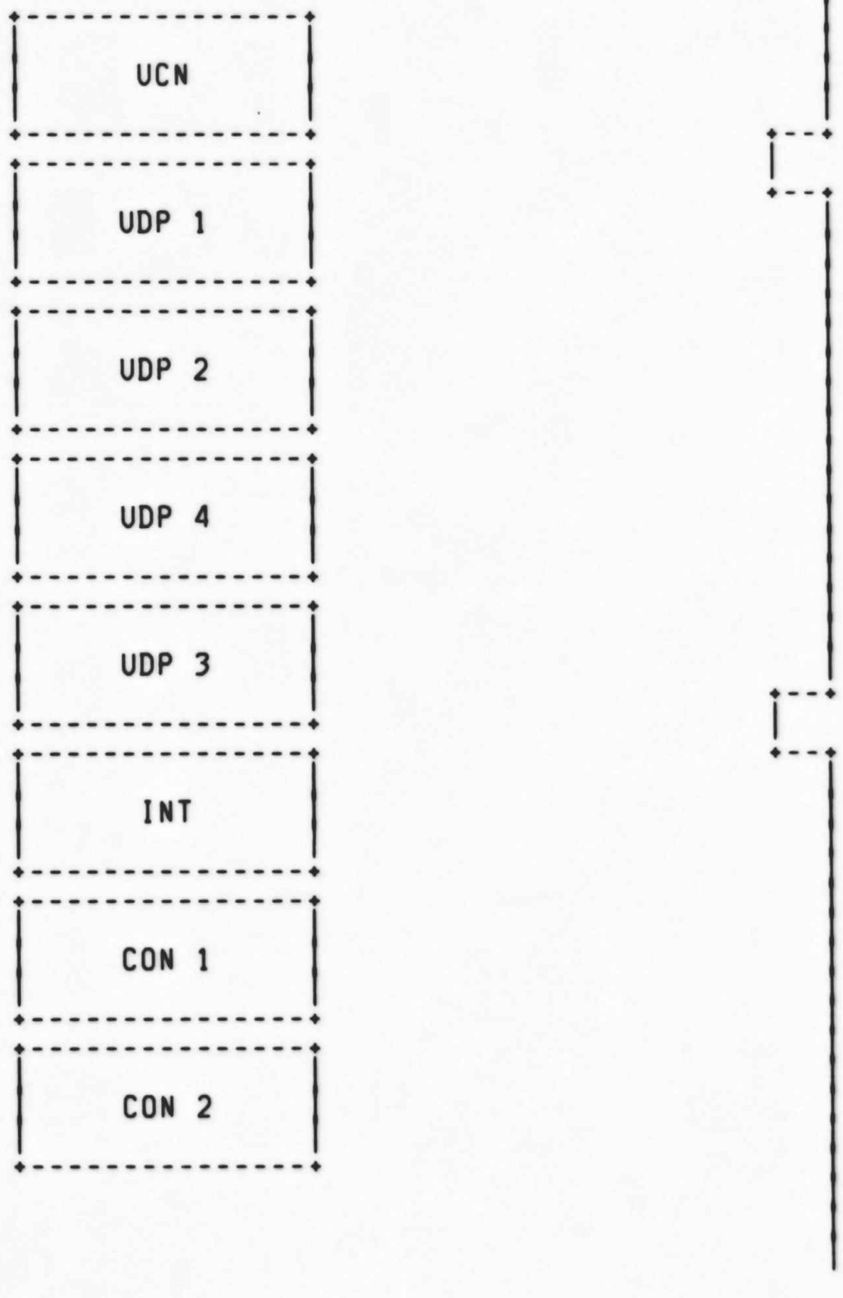
```
ZZ-ECKAB-9.2    " L0002 GATE ARRAY LOCATION"                    13-MAY-1987       Fiche 3 Frame I5          Sequence 472
ECKAB                        VAX-11/750 DPM MICRO DIAGNOSTICS     11-DEC-1985 09:34:06  VAX/VMS Macro V04-00        Page  1
V09.01                        " L0002 GATE ARRAY LOCATION"        14-AUG-1984 15:55:16  [VAX750]COMETASM.MAR;221          (1

                                0000   309 ;+-----------------------------------------------------------------------------+
```

```
                        311          .SBTTL  "       L0003 GATE ARRAY LOCATION"
                        312 ;+-----------------------------------------------------------------------------+
                        313 ;|                                                                             |
                        314 ;|                                                                             |
                        315 ;|                                                                             |
                        316 ;|                                                                             |
                        317 ;|                                                                             |
                        318 ;|                                                                             |
                        319 ;|                                                                             |
                        320 ;|                                                                             |
                        321 ;|                                                                             |
                        322 ;|                                                                             |
                        323 ;|                                                                             |
                        324 ;|                                                                             |
                        325 ;|                                                                             |
                        326 ;|                                                                             |
                        327 ;|                                                                             |
                        328 ;|                                                                             |
                        329 ;|                                                                             |
                        330 ;|                                                                             |
                        331 ;|                                                                             |
                        332 ;|                                                                             |
                        333 ;|                                                                             |
                        334 ;|                                                                             |
                        335 ;|                                                                             |
                        336 ;|                                                                             |
                        337 ;|                                                                             |
                        338 ;|                                                                             |
                        339 ;|                                                                             |
                        340 ;|                                                                             |
                        341 ;|                                                                             |
                        342 ;|                                                                             |
                        343 ;|                                                                             |
                        344 ;|                                                                             |
                        345 ;|                                                                             |
                        346 ;|                                                                             |
                        347 ;|                                                                             |
                        348 ;|                                                                             |
                        349 ;|                                                                             |
                        350 ;|                                                                             |
                        351 ;|                                                                             |
                        352 ;|                                                                             |
                        353 ;|                                                                             |
                        354 ;|                                                                             |
                        355 ;|                                                                             |
                        356 ;|                                                                             |
                        357 ;|                                                                             |
                        358 ;|                                                                             |
                        359 ;|                                                                             |
                        360 ;|                                                                             |
                        361 ;|                                                                             |
                        362 ;|                                                                             |
                        363 ;|                                                                             |
                        364 ;|                                                                             |
                        365 ;|                                                                             |
```



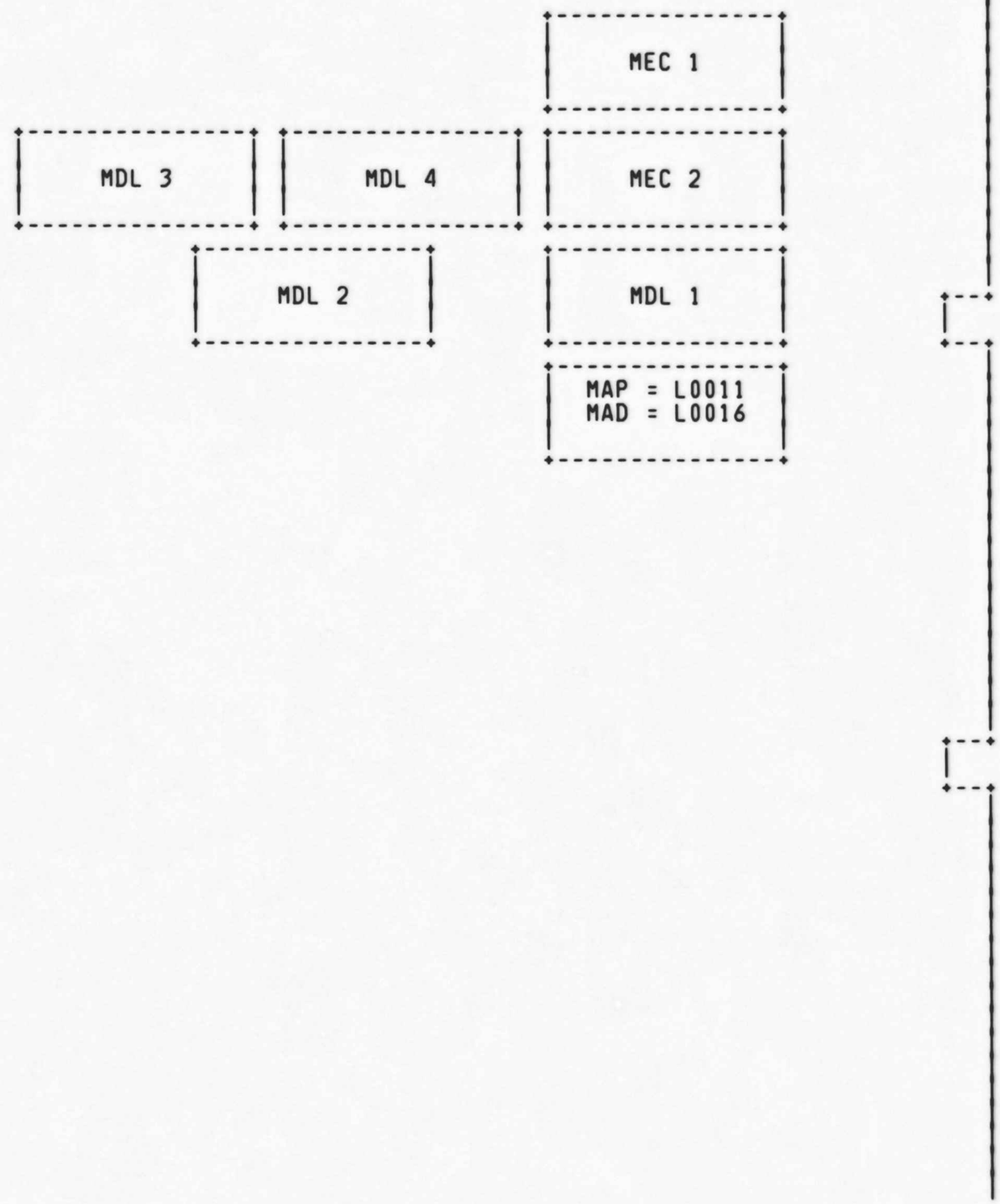
```
0000   366 ;|              +---------------+ +---------------+ +---------------+          |
0000   367 ;+-------------------------------------------------------------------------------+
```

```
                        0000    369          .SBTTL  "       L0004 GATE ARRAY LOCATION"
                        0000    370 ;+----------------------------------------------------------------------------+
                        0000    371 ;|                                                                            |
                        0000    372 ;|                                                                            |
                        0000    373 ;|                                                                            |
                        0000    374 ;|                                                                            |
                        0000    375 ;|                                                                            |
                        0000    376 ;|                                                                            |
                        0000    377 ;|                                                                            |
                        0000    378 ;|                                                                            |
                        0000    379 ;|                                                                            |
                        0000    380 ;|                                                                            |
                        0000    381 ;|                                                                            |
                        0000    382 ;|                                                                            |
                        0000    383 ;|                                                                            |
                        0000    384 ;|                                                                            |
                        0000    385 ;|                                                                            |
                        0000    386 ;|                                                                            |
                        0000    387 ;|                                                                            |
                        0000    388 ;|                                                                            |
                        0000    389 ;|                                                                            |
                        0000    390 ;|                                                                            |
                        0000    391 ;|                                                                            |
                        0000    392 ;|                                                                            |
                        0000    393 ;|                                                                            |
                        0000    394 ;|                                                                            |
                        0000    395 ;|                                                                            |
                        0000    396 ;|                                                                            |
                        0000    397 ;|                                                                            |
                        0000    398 ;|                                                                            |
                        0000    399 ;|                                                                            |
                        0000    400 ;|                                                                            |
                        0000    401 ;|                                                                            |
                        0000    402 ;|                                                                            |
                        0000    403 ;|                                                                            |
                        0000    404 ;|                                                                            |
                        0000    405 ;|                                                                            |
                        0000    406 ;|                                                                            |
                        0000    407 ;|                                                                            |
                        0000    408 ;|                                                                            |
                        0000    409 ;|                                                                            |
                        0000    410 ;|                                                                            |
                        0000    411 ;|                                                                            |
                        0000    412 ;|                                                                            |
                        0000    413 ;|                                                                            |
                        0000    414 ;|                                                                            |
                        0000    415 ;|                                                                            |
                        0000    416 ;|                                                                            |
                        0000    417 ;|                                                                            |
                        0000    418 ;|                                                                            |
                        0000    419 ;|                                                                            |
                        0000    420 ;|                                                                            |
                        0000    421 ;|                                                                            |
                        0000    422 ;|                                                                            |
                        0000    423 ;|                                                                            |
```



```
0000   424 ;|                                                                    |
0000   425 ;+-------------------------------------------------------------------+
```

```
                        .SBTTL  "       L0011 & L0016 GATE ARRAY LOCATION"
```



```
0000   482 ;|                                                                                |
0000   483 ;+--------------------------------------------------------------------------------+
```

```
                        0000    485          .SBTTL  ""
                        0000    486          .LIST   MC
                        0000    487          .LIBRARY        /UCOD$0:[VAX750]COMETMAC/
                        0000    488          .LIST   ME
                        0000    489          .NLIST  CND,MC
                        0000                 .SBTTL  "EQUATED SYMBOLS
                        0000         ;
                        0000         ; RDM REGISTER DEFINITIONS:
                        0000         ;
                        0000         ; NOTE: THESE REGITER DEFINITIONS ARE OFFSET BY THE LOAD/START ADDRESS OF
                        0000         ;       THE PROGRAM. FOR THE SIMULATOR VERSION THIS IS 400(X) AND FOR THE
                        0000         ;       REAL VERSION IT IS 8400(X).
                        0000         ;
00007400                0000                 DCS_DATA_REG     =       ^XF800 + HEAD
00007401                0000                 DCS_ADDR_REG_WO  =       ^XF801 + HEAD
00007402                0000                 DCS_CTRL_REG     =       ^XF802 + HEAD
00007403                0000                 DCS_CTRL_FILE    =       ^XF803 + HEAD
00007404                0000                 ADDR_MATCH_LO    =       ^XF804 + HEAD
00007405                0000                 ADDR_MATCH_HI    =       ^XF805 + HEAD
00007406                0000                 STATUS_REG       =       ^XF806 + HEAD
                        0000         ;
00007410                0000                 A_REG_BYTE_0     =       ^XF810 + HEAD
00007411                0000                 A_REG_BYTE_1     =       ^XF811 + HEAD
00007412                0000                 A_REG_BYTE_2     =       ^XF812 + HEAD
00007413                0000                 A_REG_BYTE_3     =       ^XF813 + HEAD
00007414                0000                 MD_REG_BYTE_0    =       ^XF814 + HEAD
00007415                0000                 MD_REG_BYTE_1    =       ^XF815 + HEAD
00007416                0000                 MD_REG_BYTE_2    =       ^XF816 + HEAD
00007417                0000                 MD_REG_BYTE_3    =       ^XF817 + HEAD
00007418                0000                 MH_REG_BYTE_0    =       ^XF818 + HEAD
00007419                0000                 MH_REG_BYTE_1    =       ^XF819 + HEAD
0000741A                0000                 MH_REG_BYTE_2    =       ^XF81A + HEAD
0000741B                0000                 MH_REG_BYTE_3    =       ^XF81B + HEAD
0000741C                0000                 CMI_CTRL_REG     =       ^XF81C + HEAD
0000741D                0000                 MAINT_REG        =       ^XF81D + HEAD
0000741E                0000                 RD_CTRL_REG      =       ^XF81E + HEAD
                        0000         ;
00007420                0000                 FRONT_PNL_1      =       ^XF820 + HEAD
00007421                0000                 FRONT_PNL_2      =       ^XF821 + HEAD
00007422                0000                 CS_ADD_TRAP_LOW  =       ^XF822 + HEAD
00007423                0000                 CS_ADD_TRAP_HGH  =       ^XF823 + HEAD
00007424                0000                 CS_ADD_BUFF_LOW  =       ^XF824 + HEAD
00007425                0000                 CS_ADD_BUFF_HGH  =       ^XF825 + HEAD
00007426                0000                 BUFF_ADDR_LOW    =       ^XF826 + HEAD
00007427                0000                 DCS_ADDR_REG_RO  =       ^XF827 + HEAD
                        0000         ;
0000742C                0000                 WD_REG_BYTE_0    =       ^XF82C + HEAD
0000742D                0000                 WD_REG_BYTE_1    =       ^XF82D + HEAD
0000742E                0000                 WD_REG_BYTE_2    =       ^XF82E + HEAD
0000742F                0000                 WD_REG_BYTE_3    =       ^XF82F + HEAD
00007430                0000                 WH_REG_BYTE_0    =       ^XF830 + HEAD
00007431                0000                 WH_REG_BYTE_1    =       ^XF831 + HEAD
00007432                0000                 WH_REG_BYTE_2    =       ^XF832 + HEAD
00007433                0000                 WH_REG_BYTE_3    =       ^XF833 + HEAD
                        0000         ;
```

```
                        0000    ; RDM REGISTER NAMES AS DEFINED IN HARDWARE SPEC
                        0000    ;
                00007400 0000           DCSDA=DCS_DATA_REG              ;Diagnostic control store data register
                00007401 0000           DCSAD=DCS_ADDR_REG_WO           ;Diagnostic control store address reg
                00007402 0000           DCSCR=DCS_CTRL_REG              ;Diagnostic control store control reg
                00007403 0000           DCSCF=DCS_CTRL_FILE             ;Diagnostic control store control file
                00007404 0000           CSAMR=ADDR_MATCH_LO             ;Control store address match register
                00007406 0000           STATUS=STATUS_REG               ;Status register
                00007410 0000           MAREG=A_REG_BYTE_0              ;Memory address register
                00007414 0000           MDREG=MD_REG_BYTE_0             ;Memory data register
                00007418 0000           MHREG=MH_REG_BYTE_0             ;Memory holding register
                0000741C 0000           CMICTL=CMI_CTRL_REG             ;CMI control
                0000741D 0000           MAIN32=MAINT_REG                ;32 Bit path maintenance control
                0000741E 0000           RDCTRL=RD_CTRL_REG              ;Remote diagnosis control
                00007420 0000           FRONT1=FRONT_PNL_1              ;Front panel readback 1
                00007421 0000           FRONT2=FRONT_PNL_2              ;Front panel readback 2
                00007422 0000           CSTRP=CS_ADD_TRAP_LOW           ;Control store address trap register
                00007424 0000           CSBUF=CS_ADD_BUFF_LOW           ;Control store address buffer
                00007426 0000           BADD1=BUFF_ADDR_LOW             ;Control store address trace readback
                00007427 0000           CLKDCS=DCS_ADDR_REG_RO          ;Clock control & DCS address register
                0000742C 0000           WDREG=WD_REG_BYTE_0             ;WBUS data register
                00007430 0000           WHREG=WH_REG_BYTE_0             ;WBUS holding register
                        0000    ;
                        0000    ; RDM REGISTER BIT DEFINITIONS:
                        0000    ;
                        0000    ; DCS CONTROL REGISTER
                00000001 0000           HALT_ON_MATCH   =       1
                00000002 0000           CLEAR_CTRL_FILE =       2
                00000004 0000           TRACE_ENABL     =       4
                00000008 0000           PAR_CHK_ENABL   =       8
                00000010 0000           PAR_STOP_ENABL  =       ^X10
                00000020 0000           CLK_CTRL_0      =       ^X20    ; ACTIVE LOW
                00000040 0000           CLK_CTRL_1      =       ^X40    ; ACTIVE LOW
                00000080 0000           MICRO_ADDR_INH  =       ^X80
                        0000        ;
                        0000        ; FOLLOWING IS A FUNCTION TABLE OF THE CLOCK CONTROL BITS
                        0000        ;
                        0000            ;       CLK_CTRL 1,0    FUNCTION
                        0000            ;           11            HALT
                        0000            ;           01            SINGLE MICRO INSTRUCTION
                        0000            ;           10            SINGLE TICK
                        0000            ;           00            RUN
                        0000    ;
                        0000    ; DCS CONTROL FILE      ALL OF THESE BITS ARE ACTIVE LOW
                        0000    ;
                00000001 0000           V_BUS_STROBE    =       1
                00000002 0000           DCS_ADDR_CLEAR  =       2
                00000004 0000           STOP_CLOCK      =       4
                00000008 0000           EN_TRACE_REG    =       8
                00000010 0000           H_FROM_WBUS     =       ^X10
                00000020 0000           WBUS_FROM_D     =       ^X20
                00000040 0000           STROBE_CMI      =       ^X40
                00000080 0000           SA_CLOCK        =       ^X80
                        0000    ;
                        0000    ; CS ADDRESS MATCH REGISTER (HIGH)
```

```
                    0000        ;
00000040            0000                VBUS_SERIAL_IN  =       ^X40    ; ACTIVE LOW
00000080            0000                VBUS_CLOCK      =       ^X80
                    0000        ;
                    0000        ; STATUS REGISTER
                    0000        ;
00000001            0000                VBUS_SERIAL_OUT =       1
00000002            0000                TRAP_OFF        =       2
00000008            0000                CMI_STATUS_0    =       8
00000010            0000                CMI_STATUS_1    =       ^X10
                    0000                ;
                    0000                ; THE CMI STATUS BITS ARE ENCODED AS FOLLOWS:
                    0000                ;
                    0000                ;       00 = NO RESPONSE
                    0000                ;       01 = UNCORRECTABLE ERROR
                    0000                ;       10 = CORRECTABLE ERROR
                    0000                ;       11 = NO ERRORS
00000020            0000                CMI_COMPLETE    =       ^X20
                    0000        ;
00000080            0000                CLOCK_STOPED    =       ^X80
                    0000        ;
                    0000        ; CMI CONTROL REGISTER
                    0000        ;
00000001            0000                CMI_CTRL_CLEAR  =       1
00000008            0000                CMI_GO          =       8
00000020            0000                CMI_WRITE       =       ^X20
00000040            0000                CMI_READ        =       ^X40
00000080            0000                SA_START_STOP   =       ^X80
                    0000        ;
                    0000        ; 32 BIT MAINTENANCE CONTROL REGISTER
                    0000        ;
00000001            0000                MAINT_TRAP_OFF  =       1
00000002            0000                MAINT_STB_INH   =       2
00000004            0000                MAINT_DCS_ENABL =       4
00000008            0000                MAINT_WB_TO_WH  =       8
00000010            0000                MAINT_WD_TO_WB  =       ^X10
00000020            0000                MAINT_CMI_TO_MH =       ^X20
00000040            0000                MAINT_MD_TO_CMI =       ^X40
00000080            0000                MAINT_A_TO_CMI  =       ^X80
                    0000        ;
                    0000        ; RD CONTROL REGISTER
                    0000        ;
00000008            0000                MASTER_HALT_EN  =       8
00000010            0000                VBUS_LOAD       =       ^X10
00000020            0000                TRAP_HALT_EN    =       ^X20
00000040            0000                DC_LOW          =       ^X40
00000080            0000                AC_LOW          =       ^X80
                    0000        ;
                    0000        ; FRONT PANEL REGISTER 1
                    0000        ;
00000001            0000                FP_BOOT_0       =       1
00000002            0000                FP_BOOT_1       =       2
00000004            0000                FP_START_0      =       4
00000008            0000                FP_START_1      =       8
00000010            0000                CPU_RUN         =       ^X10
```

```
00000020  0000                    PARITY_ERROR    =       ^X20
          0000            ;
          0000            ; FRONT PANEL REGISTER 2
          0000            ;
00000001  0000                    REMOTE          =       1
00000002  0000                    FAULT           =       2
00000004  0000                    TEST            =       4
          0000            ;
          0000            ;
          0000            ;
00000040  0000                    PB_INIT         =       ^X40    ; ACTIVE LOW
00000080  0000                    FRONT_PNL_LOCK  =       ^X80
          0000            ;
          0000            ; DCS ADDRESS REGISTER READ ONLY
          0000            ;
00000040  0000                    CLK_CTRL_0_RO   =       ^X40    ; ACTIVE HIGH
00000080  0000                    CLK_CTRL_1_RO   =       ^X80    ; ACTIVE HIGH
          0000            ;
          0000            ; MISCELLANEOUS EQUATES:
          0000            ;
00000001  0000                    BYTE            =       1
00000002  0000                    WORD            =       2
00000004  0000                    LONG            =       4
0000000A  0000                    MICROWORD       =       10
00002C5E  0000                    MONITOR_SIZE    =       ^X2C5E
00000400  0000                    STACK_SIZE      =       1024
000007A2  0000                    M$TEST_SIZE     =       14336 - STACK_SIZE - MONITOR_SIZE
          0000                                              ; MAXIMUM SIZE OF A TEST OVERLAY
000003E2  0000                    M$TEST_SIZE_D   = M$TEST_SIZE - 960
          0000                                              ; MAXIMUM SIZE OF TEST OVERLAY WITH
          0000                                              ; 'DEBUG' ENABLED IN
00000001  0000                    B               =       BYTE
00000002  0000                    W               =       WORD
00000004  0000                    L               =       LONG
0000000A  0000                    M               =       MICROWORD
00000001  0000                    HIGH            =       1
00000000  0000                    LOW             =       0
00000000  0000                    ONERROR         =       0
00000001  0000                    NOERROR         =       1
00000002  0000                    ONEQUAL         =       2
00000003  0000                    NOTEQUAL        =       3
00000010  0000                    MAX_ARGUMENTS   =       16      ; BYTE LENGTH OF LARGEST PSEUDO OP
00000036  0000                    DCS_SCRATCH_ADR =       54      ; DCS SCRATCH ADDRESS START
000032FC  0000                    ERROR_LOG_ADR   =       ^XB6FC + HEAD
          0000                                              ; START ADDRESS OF MEMORY ERROR LOG
          0000    491       .NLIST  ME
          0000    492       .LIST   MC
          0000    493       .SBTTL  ""
          0000    494       .NLIST  MC      ; SHUT-OFF LISTING FOR FIRST 'NEWTEST' CALL
```

```
                        0030        .SBTTL  "TEST   063     Rotator Test 49 Convert BCD to Numeric Test
                        0030      3 ;*******************************************************************************
                        0030      4 ;++
                        0030      5 ;
                        0030      6 ; FUNCTIONAL DESCRIPTION
                        0030      7 ;
                        0030      8 ;       This test examines the rotator's ability to convert four BCD digits on
                        0030      9 ;       MBUS to four numeric digits.
                        0030     10 ;
                        0030     11 ; TEST ALGORITHM
                        0030     12 ;
                        0030     13 ;       This test is implemented using seven 64 bit stuck at zero or one test
                        0030     14 ;       patterns. These patterns are input to the rotator from the MBUS
                        0030     15 ;       concatenated with the RBUS. They are then extracted and shuffled by
                        0030     16 ;       the convert function before being placed on the SBUS. From the SBUS
                        0030     17 ;       they are passed through the ALP's to the WBUS and the RDM's WHREG.
                        0030     18 ;       The test algorithm executes as follows:
                        0030     19 ;
                        0030     20 ;        1. Initialize the CPU.
                        0030     21 ;        2. Start signature analysis.
                        0030     22 ;        3. Load the micro-code for the first six test patterns.
                        0030     23 ;        4. Loop on I for 6 stuck at zero or one test patterns.
                        0030     24 ;        5. Generate a stuck a zero or one test pattern for a longword.
                        0030     25 ;        6. Load the pattern into the WDREG.
                        0030     26 ;        7. Loop on error from here.
                        0030     27 ;        8. Execute the micro-code to:
                        0030     28 ;               1. Load the test pattern from WDREG to MTEMP0.
                        0030     29 ;               2. Load the test pattern from WDREG to RTEMP1.
                        0030     30 ;               3. Placing RTEMP1 on the RBUS and MTEMP0 on the MBUS, execute
                        0030     31 ;                  the convert function and route the result on the SBUS
                        0030     32 ;                  through the ALP's WMUX to the RDM's WHREG.
                        0030     33 ;        9. Compare the contents of the WHREG against a table of expected data.
                        0030     34 ;       10. If the two do not match, signal an error.
                        0030     35 ;       11. Continue the loop started at step 4 for the remaining patterns.
                        0030     36 ;       12. Load the micro-code for the seventh test pattern.
                        0030     37 ;       13. Loop once on I = 7 for the last stuck at zero or one test pattern.
                        0030     38 ;       14. Loop on error from here.
                        0030     39 ;       15. Execute the micro-code to:
                        0030     40 ;               4. Load the LONLIT register with all ones.
                        0030     41 ;               5. Load MTEMP0 with zero.
                        0030     42 ;               6. Load MTEMP1 (and RTEMP1) with the LONLIT register.
                        0030     43 ;               7. Placing RTEMP1 on the RBUS and MTEMP0 on the MBUS, execute
                        0030     44 ;                  the convert function and route the result on the SBUS
                        0030     45 ;                  through the ALP's WMUX to the RDM's WHREG.
                        0030     46 ;       16. Compare the contents of the WHREG against a table of expected data.
                        0030     47 ;       17. If the two do not match, signal an error.
                        0030     48 ;       18. End the loop started at step 13.
                        0030     49 ;       19. Stop signature analysis.
                        0030     50 ;       20. End of this test, skip to the next test.
                        0030     51 ;
                        0030     52 ; TEST DATA
                        0030     53 ;
                        0030     54 ;       Seven stuck at zero or one test patterns are used to test the BCD to
                        0030     55 ;       numeric conversion. The conversion extracts and merges fields from the
                        0030     56 ;       MBUS concatenated with the RBUS and places the result on the SBUS as
```

```
                                0030     57 ;       follows:
                                0030     58 ;
                                0030     59 ;               SBUS <03:00> <-- MBUS <07:04>
                                0030     60 ;               SBUS <07:04> <-- RBUS <03:00>
                                0030     61 ;               SBUS <11:08> <-- MBUS <03:00>
                                0030     62 ;               SBUS <15:12> <-- RBUS <07:04>
                                0030     63 ;               SBUS <19:16> <-- MBUS <15:12>
                                0030     64 ;               SBUS <23:20> <-- RBUS <03:00>
                                0030     65 ;               SBUS <27:24> <-- MBUS <11:08>
                                0030     66 ;               SBUS <31:28> <-- RBUS <19:16>
                                0030     67 ;
                                0030     68 ;       The input and output data patterns used are:
                                0030     69 ;
                                0030     70 ;                  Input Data      Expected Output Data
                                0030     71 ;
                                0030     72 ;                 MBUS     RBUS          SBUS
                                0030     73 ;                AAAAAAAA AAAAAAAA      AAAAAAAA
                                0030     74 ;                55555555 55555555      55555555
                                0030     75 ;                33333333 33333333      33333333
                                0030     76 ;                0F0F0F0F 0F0F0F0F      FFF00FF0
                                0030     77 ;                00FF00FF 00FF00FF      F0F0FFFF
                                0030     78 ;                0000FFFF 0000FFFF      0FFFFFFF
                                0030     79 ;                00000000 FFFFFFFF      F0F0F0F0
                                0030     80 ;
                                0030     81 ; LOGIC DESCRIPTION
                                0030     82 ;
                                0030     83 ;       The BCD to numeric conversion is performed inside the SRM gate arrays.
                                0030     84 ;       The SRM's are controlled by the SRK gate array via the control signals
                                0030     85 ;       PRI<1:0> and SEC<5:0>.
                                0030     86 ;
                                0030     87 ; ASSUMPTIONS
                                0030     88 ;
                                0030     89 ;       This test assumes the following logic on the DPM has been tested and
                                0030     90 ;       is working; the MBUS, RBUS, SBUS and WBUS; the scratchpads RTEMP1 and
                                0030     91 ;       MTEMP0; and the ALPCTL data path for WX_S.Q_0.
                                0030     92 ;
                                0030     93 ; ERROR DESCRIPTION
                                0030     94 ;
                                0030     95 ;       On error this test will call out the SRM and SRK gate arrays. The
                                0030     96 ;       following data will also by typed:
                                0030     97 ;
                                0030     98 ;       EXPECTED SBUS DATA
                                0030     99 ;       RECEIVED SBUS DATA
                                0030    100 ;       LOOP COUNT I
                                0030    101 ;       MBUS DATA
                                0030    102 ;       RBUS DATA
                                0030    103 ;--
                                0030    104 ;*******************************************************************************
                                0030    105 ;///////////////////////////////////////////////////////////////////////////////
                                0030    106
                                0030    107         INITIALIZE                              ; INITIALIZE THE CPU
                                0032    108         LOADDCS         10$,,0,5                ; LOAD FIRST MICRO-CODE SET
                                0039    109         LOOP            I,1,6                   ; LOOP ON I FOR 6 SA01 PATTERNS
                                0040    110         SA01PAT         I,1$                    ; GENERATE A SA01 PATTERN
                                0048    111         LOADREG         WDREG,1$,,L             ; LOAD WDREG WITH PATTERN
```

```
                                  0050          ERRLOOP                                  ; LOOP ON ERROR FROM HERE
                                  0052          FETCH          0                         ; EXECUTE THE COVERSION AND
                                  0057          BRSTCLK        4                         ; WRITE THE RESULT TO WHREG
                                  005B          CMPREG         WHREG,2$,I,L              ; IS THE CONVERSION CORRECT?
                                  0064          IFERROR        2,<<SRM><SRK>>            ; IF NOT, SIGNAL AN ERROR
                                  006B          ENDLOOP        I                         ; CONTINUE LOOP
                                  006E          LOADDCS        20$,,0,6                  ; LOAD SECOND MICRO-CODE SET
                                  0075          LOOP           I,7,7                     ; LOOP ONCE FOR 7TH PATTERN
                                  007C          ERRLOOP                                  ; LOOP ON ERROR FROM HERE
                                  007E          FETCH          0                         ; EXECUTE THE CONVERSION ON 7TH
                                  0083          BRSTCLK        5                         ; PATTERN AND WRITE TO WHREG
                                  0087          CMPREG         WHREG,2$,I,L              ; IS THE CONVERSION CORRECT?
                                  0090          IFERROR        2,<<SRM><SRK>>            ; IF NOT, SIGNAL AN ERROR
                                  0097          ENDLOOP        I                         ; END OF LOOP FOR 7TH PATTERN
                                  009A          SKIP                                     ; END OF TEST
                                  00A1
                                  00A1    BEGIN_TEST_DATA
                                  00A1
                                  00A1    ;-----------------------------------------------------------------------
                                  00A1
                        00000000  00A1    1$:     .LONG   0       ; LOCATION TO HOLD SA01 PATTERNS
                                  00A5
                                  00A5    ;
                                  00A5    ; Expected Data
                                  00A5    ;
                                  00A5
                        AAAAAAAA  00A5    2$:     .LONG   ^XAAAAAAAA
                        55555555  00A9            .LONG   ^X55555555
                        33333333  00AD            .LONG   ^X33333333
                        FFF00FF0  00B1            .LONG   ^XFFF00FF0
                        F0F0FFFF  00B5            .LONG   ^XF0F0FFFF
                        0FFFFFFF  00B9            .LONG   ^X0FFFFFFF
                        F0F0F0F0  00BD            .LONG   ^XF0F0F0F0
                                  00C1
                                  00C1    ;
                                  00C1    ; Test micro-code sets
                                  00C1    ;
                                  00C1
                                  00C1    10$:
U 00, 7700,C800,0364,0300,0470,1801  00C1    ;% 00   NOP
U 01, 5700,C860,0364,0300,0470,1802  00CC    ;% 01   M[TEMP0]_RDM
U 02, 5700,4840,0364,0304,0470,1803  00D7    ;% 02   R[TEMP1]_RDM
U 03, 6700,C800,6F47,0304,0470,1804  00E2    ;% 03   RSRC/TEMP1,MSRC/TEMP0,ROT/CVTPN,ALPCTL/WX_S.Q_0,RDM_WB
U 04, 7300,4800,0364,0300,0470,1805  00ED    ;% 04   STOP.CLOCK
                                  00F8
                                  00F8    20$:
U 00, 7700,C800,0364,0300,0470,1801  00F8    ;% 00   NOP
U 01, 7700,B800,0000,0000,0470,1802  0103    ;% 01   LONLIT_[0FFFFFFFF]
U 02, 7700,4860,5BE4,03D8,0470,1803  010E    ;% 02   M[TEMP0]_R[ZERO]
U 03, 7700,8861,5BE4,03D4,0470,1804  0119    ;% 03   M[TEMP1]_R[LONLIT]
U 04, 6700,4800,6F47,0304,0470,1805  0124    ;% 04   RSRC/TEMP1,MSRC/TEMP0,ROT/CVTPN,ALPCTL/WX_S.Q_0,RDM_WB
U 05, 7300,4800,0364,0300,0470,1806  012F    ;% 05   STOP.CLOCK
                                  013A
                                  013A    END_TEST_DATA
                                  013A
```

```
013A        ;-----------------------------------------------------------------------
013A    197
013A    198 ENDTEST
```

```
001F        .SBTTL  "TEST   064     Control store address test
001F    ;*****************************************************************************
001F    ;++
001F    ;
001F    ; FUNCTIONAL DESCRIPTION
001F    ;
001F    ;       This test checks the control store address for open or shorted lines.
001F    ;       This test has been divided into two subtests because different logic
001F    ;       is involved between bits <12:06> and <05:00> of the control store
001F    ;       address.
001F    ;
001F    ; TEST ALGORITHM
001F    ;
001F    ;   Subtest #1
001F    ;
001F    ;       This subtest examines bits <05:00> of the control store address logic.
001F    ;
001F    ;       1.  Initialize the CPU.
001F    ;       2.  Prepare a loop of eight iterations.
001F    ;       3.  Load the Next Address field of the test micro instruction with a
001F    ;           test pattern read from table one for the current loop iteration.
001F    ;       4.  Load the test micro-instruction into DCS.
001F    ;       5.  Execute the micro-instruction:
001F    ;               1. NOP,NEXT/<test pattern>,STOP.CLOCK
001F    ;       6.  Do a masked compare for bits <05:00> of the control store address
001F    ;           lines from the RDM address trace register with the expected value
001F    ;           stored in RDM memory.
001F    ;       7.  If the two are different signal an error.
001F    ;       8.  Continue steps 3-6 until all eight test patterns have been used.
001F    ;
001F    ;   Subtest #2
001F    ;
001F    ;       This subtest examines bits <12:06> of the control store address logic.
001F    ;
001F    ;       9.  Prepare a second loop of eight iterations.
001F    ;       10. Load the Next Address field of the test micro-instruction with a
001F    ;           test pattern read from table two, indexed by the loop counter.
001F    ;       11. Load the modified test micro-instruction into DCS.
001F    ;       12. Execute the micro-instruction:
001F    ;               1. NOP,NEXT/<test pattern>,STOP.CLOCK
001F    ;       13. Do a masked compare for bits <12:06> of the control store address
001F    ;           lines from the RDM address trace register with the expected value
001F    ;           stored in RDM memory.
001F    ;       14. If the two are different signal an error.
001F    ;       15. Continue steps 9-12 until all eight test patterns have been used.
001F    ;       16. Test completed.
001F    ;
001F    ; DATA PATTERNS
001F    ;
001F    ;       TABLE 1         MASK = 003F
001F    ;
001F    ;       Loop Count              Control Store Address Data
001F    ;
001F    ;           1                           1 F F F
001F    ;           2.                          1 F C 0
```

```
001F    254 ;           3.                          1 F F E
001F    255 ;           4.                          1 F F D
001F    256 ;           5.                          1 F F B
001F    257 ;           6.                          1 F F 7
001F    258 ;           7.                          1 F E F
001F    259 ;           8.                          1 F D F
001F    260 ;
001F    261 ;       TABLE 2         MASK = 1FC0
001F    262 ;
001F    263 ;       Loop Count                  Control Store Address Data
001F    264 ;
001F    265 ;           1.                          1 F F F
001F    266 ;           2.                          0 0 3 F
001F    267 ;           3.                          1 F B F
001F    268 ;           4.                          1 F 7 F
001F    269 ;           5.                          1 E F F
001F    270 ;           6.                          1 D F F
001F    271 ;           7.                          1 B F F
001F    272 ;           8.                          1 7 F F
001F    273 ;           9.                          0 F F F
001F    274 ;
001F    275 ;
001F    276 ; LOGIC DESCRIPTION
001F    277 ;
001F    278 ;       In subtest 1 the MSQ, PHB and IRD gate arrays have open collector
001F    279 ;       outputs that are tied together on the lower six bits <05:00> of the
001F    280 ;       control store address. MSQ and PHB are tied together on all six bits,
001F    281 ;       but the IRD gate array is only tied to bits <03:00>. Thus when this
001F    282 ;       test first detects an error the MSQ gate array should be replaced and
001F    283 ;       then the test run again. If this test fails a second time you can
001F    284 ;       compare the expected data to the received data to give you a good clue
001F    285 ;       as to whether the PHB or IRD chip might be causing the problem. If all
001F    286 ;       three gate arrays are replaced and the test still fails then discrete
001F    287 ;       logic on the DPM is suspect.
001F    288 ;
001F    289 ;       In subtest 2 the high order eight bits of the control store address
001F    290 ;       do not go through any gate arrays. Thus if an error occurs in one of
001F    291 ;       these address lines the problem must be traced down to discrete logic
001F    292 ;       on the DPM.
001F    293 ;
001F    294 ; ASSUMPTIONS
001F    295 ;
001F    296 ;       This test requires no signals from any other CPU modules.
001F    297 ;
001F    298 ; ERROR DESCRIPTION
001F    299 ;
001F    300 ;       Subtest #1
001F    301 ;
001F    302 ;       On error subtest 1 will call out the MSQ, PHB and IRD gate arrays on
001F    303 ;       the DPM module.
001F    304 ;
001F    305 ;       Subtest #2
001F    306 ;
001F    307 ;       On error subtest 2 will call out only the DPM module since no gate
001F    308 ;       arrays are attached to bits <12:06> of the control store address.
```

```
                     001F    309 ;
                     001F    310 ;       On error both subtests will also type out the following information:
                     001F    311 ;
                     001F    312 ;       Expected Data:
                     001F    313 ;       Received Data:
                     001F    314 ;       Loop Count:
                     001F    315 ;
                     001F    316 ;--
                     001F    317 ;*****************************************************************************
                     001F    318
                     001F    319         SUBTEST         1
                     0022
                     0022        ;////////////////////////////////////////////////////////////////////////////
                     0022    320 ;+
                     0022    321 ; Subtest 1
                     0022    322 ;
                     0022    323 ;       This subtest examines bits <05:00> of the control store address.
                     0022    324 ;-
                     0022    325         INITIALIZE                                  ; INITIALIZE THE CPU
                     0024    326         LOOP            I,1,8                       ; LOOP FOR 8 ITERATIONS
                     002B    327         LDFIELD         3$,I,WORD,2$,0,13           ; MODIFY THE NEXT ADDRESS FIELD
                     003C    328         LOADDCS         1$,,0,3                     ; LOAD THE MODIFIED INSTRUCTION
                     0043    329         EXPUTRAP                                    ; PREVENT UNEXPECTED CLOCK STOP
                     0045    330         ERRLOOP                                     ; ERROR LOOP HERE
                     0047    331         INITIALIZE                                  ; CLEAR OUT CSPE U-TRAP
                     0049    332         FETCH           0                           ; FLOAT A ZERO THROUGH BITS
                     004E    333         BRSTCLK         2                           ; <05:00> OF THE CS ADDRESS
                     0052    334         CMPREGMSK       CSTRP,3$,I,5$,,WORD,        ; DOES CSTRP = EXPECTED ADDRESS?
                     005E    335         IFERROR         ,<<MSQ>,<PHB>,<IRD>>        ; IF NOT SIGNAL ERROR
                     0066    336         ENDLOOP         I                           ; CONTINUE LOOP FOR 8 PATTERNS
                     0069    337
                     0069    338         SUBTEST         2
                     006C
                     006C        ;////////////////////////////////////////////////////////////////////////////
                     006C    339 ;+
                     006C    340 ; Subtest 2
                     006C    341 ;
                     006C    342 ;       This subtests examines bits <12:06> of the control store address.
                     006C    343 ;-
                     006C    344         LOOP            I,1,9                       ; LOOP FOR 8 TEST PATTERNS
                     0073    345         LDFIELD         4$,I,WORD,2$,0,13           ; MODIFY THE NEXT ADDRESS FIELD
                     0084    346         LOADDCS         1$,,0,3                     ; LOAD THE MODIFIED U-INST.
                     008B    347         EXPUTRAP                                    ; PREVENT UNEXPECTED CLOCK STOP
                     008D    348         ERRLOOP                                     ; ERROR LOOP HERE
                     008F    349         INITIALIZE                                  ; CLEAR OUT CSPE U-TRAP
                     0091    350         FETCH           0                           ; FLOAT A ZERO THROUGH BITS
                     0096    351         BRSTCLK         2                           ; <12:06> OF THE CS ADDRESS
                     009A    352         CMPREGMSK       CSTRP,4$,I,6$,,WORD         ; DOES CSTRP = EXPECTED ADDRESS?
                     00A6    353         IFERROR         ,,<<DPM>>                   ; IF NOT SIGNAL AN ERROR
                     00AC    354         ENDLOOP         I                           ; CONTINUE FOR 8 TEST PATTERNS
                     00AF    355         SKIP                                        ; END OF THIS TEST
                     00B6    356
                     00B6    357         BEGIN_TEST_DATA
                     00B6
                     00B6        ;-------------------------------------------------------------------------
```

```
                                00B6   358
                                00B6   359 1$:
U 00, 7700,C800,0364,0300,0470,1801   00B6   360 ;% 00   NOP                     ; GET CPU UP TO SPEED
                                00C1   364 2$:
U 01, 7700,C800,0364,0300,0470,1802   00C1   365 ;% 01   NOP                     ; HAS A MODIFIED NEXT ADDRESS FIELD
U 02, 7300,4800,0364,0300,0470,1803   00CC   369 ;% 02   STOP.CLOCK
                                00D7   373
                           1FFF 00D7   374 3$:     .WORD   ^X 1FFF         ; DATA PATTERNS FOR SUBTEST 1
                           1FC0 00D9   375         .WORD   ^X 1FC0
                           1FFE 00DB   376         .WORD   ^X 1FFE
                           1FFD 00DD   377         .WORD   ^X 1FFD
                           1FFB 00DF   378         .WORD   ^X 1FFB
                           1FF7 00E1   379         .WORD   ^X 1FF7
                           1FEF 00E3   380         .WORD   ^X 1FEF
                           1FDF 00E5   381         .WORD   ^X 1FDF
                                00E7   382
                           1FFF 00E7   383 4$:     .WORD   ^X 1FFF         ; DATA PATTERNS FOR SUBTEST 2
                           003F 00E9   384         .WORD   ^X 003F
                           1FBF 00EB   385         .WORD   ^X 1FBF
                           1F7F 00ED   386         .WORD   ^X 1F7F
                           1EFF 00EF   387         .WORD   ^X 1EFF
                           1DFF 00F1   388         .WORD   ^X 1DFF
                           1BFF 00F3   389         .WORD   ^X 1BFF
                           17FF 00F5   390         .WORD   ^X 17FF
                           0FFF 00F7   391         .WORD   ^X 0FFF
                                00F9   392
                           003F 00F9   393 5$:     .WORD   ^X 003F         ; MASK FOR SUBTEST 1
                           1FC0 00FB   394 6$:     .WORD   ^X 1FC0         ; MASK FOR SUBTEST 2
                                00FD   395
                                00FD   396 END_TEST_DATA
                                00FD
                                00FD       ;-----------------------------------------------------------------------------
                                00FD   397
                                00FD   398 ENDTEST
```

```
                        .SBTTL  "TEST   065     Micro Sequencer Stack Test
001F    400 ;*****************************************************************************
001F    401 ;++
001F    402 ;
001F    403 ; FUNCTIONAL DESCRIPTION
001F    404 ;
001F    405 ;         This test will examine the micro stack logic on the DPM. This test
001F    406 ;       contains two subtests, the first examines the data integrity of each
001F    407 ;       stack location while the second will check to make sure all 16 micro-
001F    408 ;       stack locations are tested.
001F    409 ;
001F    410 ; TEST ALGORITHM
001F    411 ;
001F    412 ;   Subtest #1
001F    413 ;
001F    414 ;       1.  Initialize the CPU.
001F    415 ;       2.  Prepare a loop of 16 iterations.
001F    416 ;       3.  Prepare a second loop of 6 iterations.
001F    417 ;       4.  Load the next address field of the JSR micro-instruction with a
001F    418 ;           test data pattern from Table 1 indexed by the loop counter.
001F    419 ;       5.  Load the micro-instruction at the proper location in DCS.
001F    420 ;       6.  Execute the micro-instructions:
001F    421 ;                1. NOP
001F    422 ;                2. NOP,NEXT/<test pattern>
001F    423 ;                3. JSR/PUSH
001F    424 ;                4. BUT/RETURN,NEXT/0000
001F    425 ;                5. STOP.CLOCK
001F    426 ;       7.  Compare the result in the RDM address trace register with the
001F    427 ;           expected data stored in the RDM.
001F    428 ;       8.  If the two are different signal an error.
001F    429 ;       9.  Continue steps 4-7 until all 6 test patterns have been used.
001F    430 ;       10. Load the next address field of a JSR micro-instruction with the
001F    431 ;           data pattern from Table 2 indexed by the loop counter.
001F    432 ;       11. Load the micro-instruction at the proper location in DCS.
001F    433 ;       12. Execute the micro-instructions:
001F    434 ;                5. NOP
001F    435 ;                6. NOP,NEXT/<test pattern>
001F    436 ;                7. JSR/PUSH
001F    437 ;                8. STOP.CLOCK
001F    438 ;       13. Continue steps 3-11 until all 16 stack locations have been pushed.
001F    439 ;
001F    440 ;   Subtest #2
001F    441 ;
001F    442 ;       13. Prepare a second loop of 16 iterations.
001F    443 ;       14. Execute the micro-instructions:
001F    444 ;                9. NOP
001F    445 ;               10. BUT/RETURN,NEXT/0000
001F    446 ;               11. STOP.CLOCK
001F    447 ;       15. Compare the result in the Control Store Address Latch with the
001F    448 ;           original data pushed onto the stack.
001F    449 ;       16. If the two are different signal an error.
001F    450 ;       17. Continue steps 14 and 15 until all 16 addresses have been popped
001F    451 ;           off the micro stack.
001F    452 ;       18. Test completed.
001F    453 ;
```

```
001F    454 ;
001F    455 ; TEST DATA
001F    456 ;
001F    457 ;       TABLE 1
001F    458 ;
001F    459 ;       The patterns used for Subtest #1 are:
001F    460 ;
001F    461 ;               Loop Count              Stack Data
001F    462 ;
001F    463 ;                   1.                  1 A A A
001F    464 ;                   2.                  1 D 5 5
001F    465 ;                   3.                  1 B 3 3
001F    466 ;                   4.                  1 F 0 F
001F    467 ;                   5.                  1 8 F F
001F    468 ;
001F    469 ;       TABLE 2
001F    470 ;
001F    471 ;       The following addresses are used for Subtest #2:
001F    472 ;
001F    473 ;                           Loop Count      Address Pushed Onto Micro Stack
001F    474 ;
001F    475 ;           1.                  1                   1 8 0 0
001F    476 ;           2.                  2                   1 C 4 5
001F    477 ;           3.                  3                   1 8 8 A
001F    478 ;           4.                  4                   1 C C F
001F    479 ;           5.                  5                   1 9 1 0
001F    480 ;           6.                  6                   1 D 5 5
001F    481 ;           7.                  7                   1 9 9 A
001F    482 ;           8.                  8                   1 D D F
001F    483 ;           9.                  9                   1 A 2 0
001F    484 ;          10.                  A                   1 E 6 5
001F    485 ;          11.                  B                   1 A A A
001F    486 ;          12.                  C                   1 E E F
001F    487 ;          13.                  D                   1 B 3 0
001F    488 ;          14.                  E                   1 F 7 5
001F    489 ;          15.                  F                   1 B B A
001F    490 ;          16.                 10                   1 F F F
001F    491 ;
001F    492 ;
001F    493 ; LOGIC DESCRIPTION
001F    494 ;
001F    495 ;       This test examines the following discrete logic on the DPM:
001F    496 ;
001F    497 ;       Subtest 1.  Micro-stack as implemented by four 16X4 RAMs on the DPM.
001F    498 ;       Subtest 2.  Micro-stack address lines <10:00> on the DPM.
001F    499 ;
001F    500 ; ASSUMPTIONS
001F    501 ;
001F    502 ;       This test requires no signals from any other CPU modules.
001F    503 ;
001F    504 ; ERROR DESCRIPTION
001F    505 ;
001F    506 ;       On an error in Subtest 1 & 3 only the DPM will be called out. On an
001F    507 ;       error in Subtest 2 the MSQ gate array and the DPM will be called out.
001F    508 ;
```

```
                        509 ;       On error both subtests will also type out the following information:
                        510 ;
                        511 ;       Expected Data:
                        512 ;       Received Data:
                        513 ;       Index I:
                        514 ;
                        515 ;       Subtest 2 will also type out Index J.
                        516 ;--
                        517 ;*******************************************************************************
                        518
001F                    519         SUBTEST         1
0022                    
0022                    ;////////////////////////////////////////////////////////////////////////////////
0022                    520 ;+
0022                    521 ; Subtest 1
0022                    522 ;
0022                    523 ;       This subtest examines the data integrity of each micro-stack location.
0022                    524 ;-
0022                    525
0022                    526         INITIALIZE                                  ; initialize the cpu
0024                    527         LOADDCS         5$,,0,12                    ; load test u-code
002B                    528         LOOP            I,1,16                      ; loop 1 tests all locations
0032                    529         LOOP            J,1,5                       ; loop 2 tests each location
0039                    530         LDFIELD         1$,J,WORD,6$,0,13           ; load the next address field
004A                    531         LOADDCS         6$,,1,1                     ; with a test pattern
0051                    532         EXPUTRAP                                    ; prevent unexpected clock stop
0053                    533         FETCH           0                           ; write then read each stack
0058                    534         BRSTCLK         4,INH                       ; location
005C                    535         CMPREGMSK       CSTRP,1$,J,3$,,WORD,        ; does cstrp = test pattern?
0068                    536         IFERROR         ,,<DPM>                     ; if not call out error
006E                    537         ENDLOOP         J                           ; continue for 6 test patterns
0071                    538         LDFIELD         2$,I,WORD,7$,0,13           ; load the next address field
0082                    539         LOADDCS         7$,,6,1                     ; with a test pattern and then
0089                    540         FETCH           5                           ; push this onto the stack to
008E                    541         BRSTCLK         8,INH                       ; gain access to all 16 stack
0092                    542         ENDLOOP         I                           ; locations
0095                    543
0095                    544         SUBTEST         2
0098                    
0098                    ;////////////////////////////////////////////////////////////////////////////////
0098                    545 ;+
0098                    546 ; Subtest 2
0098                    547 ;
0098                    548 ;       This subtest tests the micro-stack addressing logic to insure that all
0098                    549 ;       16 locations have been tested.
0098                    550 ;-
0098                    551
0098                    552         LOOP            I,16,1                      ; loop to read all 16 locations
009F                    553         FETCH           9                           ; read by executing but/return
00A4                    554         BRSTCLK         <^X0B>,INH                  ; with next/0000
00A8                    555         CMPREGMSK       CSTRP,2$,I,3$,,WORD         ; does cstrp = count pattern?
00B4                    556         IFERROR         ,<MSQ>,<DPM>                ; if not signal an error
00BB                    557         ENDLOOP         I                           ; continue for 16 locations
00BE                    558         SKIP                                        ; go to next test
00C5                    559
```

```
                        00C5   560
                        00C5   561          BEGIN_TEST_DATA
                        00C5
                        00C5          ;-------------------------------------------------------------------------
                        00C5   562
                1AAA    00C5   563 1$:      .WORD    ^X 1AAA                    ; Micro-stack data integrity test table
                1D55    00C7   564          .WORD    ^X 1D55
                1B33    00C9   565          .WORD    ^X 1B33
                1F0F    00CB   566          .WORD    ^X 1F0F
                18FF    00CD   567          .WORD    ^X 18FF
                        00CF   568
                1800    00CF   569 2$:      .WORD    ^X 1800                    ; Micro-stack address test table
                1C45    00D1   570          .WORD    ^X 1C45
                188A    00D3   571          .WORD    ^X 188A
                1CCF    00D5   572          .WORD    ^X 1CCF
                1910    00D7   573          .WORD    ^X 1910
                1D55    00D9   574          .WORD    ^X 1D55
                199A    00DB   575          .WORD    ^X 199A
                1DDF    00DD   576          .WORD    ^X 1DDF
                1A20    00DF   577          .WORD    ^X 1A20
                1E65    00E1   578          .WORD    ^X 1E65
                1AAA    00E3   579          .WORD    ^X 1AAA
                1EEF    00E5   580          .WORD    ^X 1EEF
                1B30    00E7   581          .WORD    ^X 1B30
                1F75    00E9   582          .WORD    ^X 1F75
                1BBA    00EB   583          .WORD    ^X 1BBA
                1FFF    00ED   584          .WORD    ^X 1FFF
                        00EF   585
                1FFF    00EF   586 3$:      .WORD    ^X 1FFF                    ; mask for stack comparisons
                        00F1   587
                        00F1   588 5$:
U 00, 7700,C800,0364,0300,0470,1801   00F1   589 ;% 00   NOP                     ; pause to get cpu up to speed
                        00FC   593 6$:
U 01, 7700,C800,0364,0300,0470,1802   00FC   594 ;% 01   NOP
U 02, 7700,C800,0364,0300,0470,5803   0107   598 ;% 02   JSR/PUSH               ; Push NEXT ADDRESS onto u-stack
U 03, 7700,0800,0364,0B00,0470,0000   0112   602 ;% 03   BUT/RETURN,NEXT/0000   ; Pop CS ADDRESS off u-stack
U 04, 7300,4800,0364,0300,0470,1805   011D   606 ;% 04   STOP.CLOCK             ; HALT
U 05, 7700,4800,0364,0300,0470,1806   0128   610 ;% 05   NOP                    ; Pause to get CPU up to speed
                        0133   614 7$:
U 06, 7700,C800,0364,0300,0470,1807   0133   615 ;% 06   NOP
U 07, 7700,4800,0364,0300,0470,5808   013E   619 ;% 07   JSR/PUSH               ; Push address onto u-stack
U 08, 7300,4800,0364,0300,0470,1809   0149   623 ;% 08   STOP.CLOCK             ; Halt
U 09, 7700,4800,0364,0300,0470,180A   0154   627 ;% 09   NOP                    ; Pause to get CPU up to speed
U 0A, 7700,0800,0364,0B00,0470,0000   015F   631 ;% 0A   BUT/RETURN,NEXT/0000   ; Pop CS ADDRESS off u-stack
U 0B, 7300,4800,0364,0300,0470,180C   016A   635 ;% 0B   STOP.CLOCK             ; HALT
                        0175   639
                        0175   640
                        0175   641 END_TEST_DATA
                        0175
                        0175          ;-------------------------------------------------------------------------
                        0175   642 ENDTEST
```

```
0024                    .SBTTL  "TEST   066     D CLOCK INHIBIT ON BUT/RET.DINH
0024    644 ;*****************************************************************************
0024    645 ;++
0024    646 ;
0024    647 ; FUNCTIONAL DESCRIPTION:
0024    648 ;
0024    649 ;       THIS TEST VERIFIES THAT D CLOCK IS INHIBITED ON A BUT/RET.DINH MICRO
0024    650 ;       ORDER.
0024    651 ;
0024    652 ;
0024    653 ; TEST ALGORITHM:
0024    654 ;
0024    655 ;       1.  EXECUTE DCS PROGRAM
0024    656 ;               a.  ZERO RTEMP0
0024    657 ;               b.  PERFORM A JSR TO PUSH AN ADDRESS ON THE MICRO STACK
0024    658 ;               c.  PERFORM A BUT/RET.DINH
0024    659 ;               d.  TRY TO WRITE -1 TO RTEMP0
0024    660 ;               e.  READ RTEMP0 TO THE RDM
0024    661 ;       2.  TEST THE VBUS FOR RTUT DINH
0024    662 ;       3.  IF NO ERROR TEST RTEMP0 FOR ZEROS
0024    663 ;       4.  IF NO ERROR GO TO NEXT TEST
0024    664 ;
0024    665 ;
0024    666 ; TEST PATTERNS:
0024    667 ;
0024    668 ;       NONE
0024    669 ;
0024    670 ;
0024    671 ; LOGIC DESCRIPTION:
0024    672 ;
0024    673 ;       THIS TEST CHECKS THAT THE BUT/RET.DINH MICRO ORDER DOES INFACT BLOCK
0024    674 ;       D CLOCK FROM OCCURRING.  IF D CLOCK WHERE NOT BLOCKED THEN RTEMP0
0024    675 ;       WOULD BE OVERWRITTEN WITH -1.
0024    676 ;
0024    677 ;
0024    678 ; ASSUMPTIONS:
0024    679 ;
0024    680 ;       THIS TEST ASSUMES THAT ADDRESSES CAN BE PUSHED AND POPPED OFF THE MICRO
0024    681 ;       STACK.
0024    682 ;
0024    683 ;
0024    684 ; ERROR DESCRIPTION:
0024    685 ;
0024    686 ;       FIRST IFERROR:
0024    687 ;               EXPECTED VBUS DATA
0024    688 ;               RECEIVED VBUS DATA
0024    689 ;                       (BITS <7:0> = VBUS BIT POSITION)
0024    690 ;                       (BITS <8> = BIT STATE)
0024    691 ;
0024    692 ;
0024    693 ;       SECOND IFERROR:
0024    694 ;               EXPECTED RTEMP0
0024    695 ;               RECEIVED RTEMP0
0024    696 ;
0024    697 ;--
```

```
                                  0024   698 ;*****************************************************************************
                                  0024   699
                                  0024   700
                                  0024   701         INITIALIZE                                  ;INIT THE CPU
                                  0026   702         ERRLOOP                                     ;ERROR LOOP POINT
                                  0028   703         FETCH          0                            ;START DCS PROGRAM
                                  002D   704         BRSTCLK        6                            ;END DCS PROGRAM
                                  0031   705         CMPVBUS        1$                           ;TEST FOR RTUT DINH ON VBUS
                                  0036   706         IFERROR        ,,<UBI>                      ;SIGNAL MISSING
                                  003C   707         CMPREG         WHREG,2$,,L,                 ;TEST RTEMP0
                                  0045   708         IFERROR        ,<SAC>,                      ;D CLOCK NOT INHIBITED
                                  004B   709         SKIP                                        ;GO TO NEXT TEST
                                  0052   710
                                  0052   711
                                  0052   712 BEGIN_TEST_DATA
                                  0052
                                  0052       ;-----------------------------------------------------------------------
                                  0052   713
                             01   0052   714 1$:     .BYTE   ^X01
                                  0053   715         VBUSDATA       <^X03>,0                     ;RTUT DINH INH ON VBUS
                       00000000   0054   716 2$:     .LONG   ^X00000000                          ;RTEMP0 CONTENTS
                                  0058   717
                                  0058   718
                                  0058   719 BEGIN_CPU_DATA
                                  0002   720
                                  0002   721 DCS_DATA
                                  0001   722
U 00, 7700,C800,0364,0300,0470,1801  0001   723 ;% 00   NOP
U 01, 7700,C840,5B70,0300,0470,1802  000C   727 ;% 01   R[TEMP0]_0                                  ;ZERO RTEMP0
U 02, 7700,C800,0364,0300,0470,5803  0017   731 ;% 02   JSR/PUSH                                    ;PUSH DCS ADDRESS ON STACK
U 03, 7700,C800,0364,0F00,0470,1804  0022   735 ;% 03   BUT/RET.DINH                                ;RETURN WITH D CLOCK INHIBIT
U 04, 7600,0840,E770,0300,0470,1805  002D   739 ;% 04   R[TEMP0]_-1,STROBE.V.BUS                    ;TRY TO WRITE -1 TO RTEMP0
U 05, 6700,4800,5BE4,0300,0470,1806  0038   743 ;% 05   RDM_WB,WB_R[TEMP0]                          ;READ RTEMP0 TO THE RDM
U 06, 7300,C800,0364,0300,0470,1807  0043   747 ;% 06   NOP,STOP.CLOCK                              ;STOP THE CPU CLOCK
                                  004E   751
                                  004E   752 END_CPU_DATA
                                  0058   753
                                  0058   754
                                  0058   755 END_TEST_DATA
                                  0058
                                  0058       ;-----------------------------------------------------------------------
                                  0058   756 ENDTEST
```

```
                        0013        .SBTTL  "TEST   067     MSQ adder test
                        0013    758 ;*****************************************************************************
                        0013    759 ;++
                        0013    760 ;
                        0013    761 ; FUNCTIONAL DESCRIPTION
                        0013    762 ;
                        0013    763 ;       This test will examine the MSQ adder used to execute micro-subroutine
                        0013    764 ;       returns.
                        0013    765 ;
                        0013    766 ; TEST ALGORITHM
                        0013    767 ;
                        0013    768 ;         1. Initialize the CPU.
                        0013    769 ;         2. Prepare a loop of 13 iterations.
                        0013    770 ;         3. Load the Next Address field of the second micro-instruction in DCS
                        0013    771 ;            with an address selected from data table one by the loop counter.
                        0013    772 ;         4. Load the Next Address field of the third micro-instruction in DCS
                        0013    773 ;            with an address selected from data table two by the loop counter.
                        0013    774 ;         5. Load the modified micro-instruction into DCS.
                        0013    775 ;         6. Execute the micro-instructions:
                        0013    776 ;                 1. NOP,NEXT/<test pattern #1>
                        0013    777 ;                 2. JSR/PUSH
                        0013    778 ;                 3. BUT/RETURN,NEXT/<test pattern #2>,STOP.CLOCK
                        0013    779 ;         7. Do a masked comparison on the resulting control store address
                        0013    780 ;            stored in the Control Store Address Trace Register with the
                        0013    781 ;            expected result from data table 3.
                        0013    782 ;         8. If the two are different signal an error.
                        0013    783 ;         9. Otherwise continue steps 3-7 until all 13 test patterns have been
                        0013    784 ;            used.
                        0013    785 ;        10. Test completed.
                        0013    786 ;
                        0013    787 ; TEST DATA
                        0013    788 ;
                        0013    789 ;       The following 39 words are stored in three data tables of 13 entries
                        0013    790 ;       each. These specific patterns were determined by the logic fault
                        0013    791 ;       simulator XLOG, to be the optimal set of patterns required to fully
                        0013    792 ;       test the adder in the MSQ gate array. They all begin with hex 18 to
                        0013    793 ;       insure that the CPU remains in DCS. The actual test patterns are in
                        0013    794 ;       the lower byte of each pattern.
                        0013    795 ;
                        0013    796 ;       Loop Count      Table 1         Table 2         Table 3
                        0013    797 ;
                        0013    798 ;            1.         1 8 0 0         1 8 3 F         1 8 3 F
                        0013    799 ;            2.         1 8 3 F         1 8 0 1         1 8 0 0
                        0013    800 ;            3.         1 8 0 A         1 8 0 A         1 8 1 4
                        0013    801 ;            4.         1 8 3 5         1 8 3 5         1 8 2 A
                        0013    802 ;            5.         1 8 0 3         1 8 1 E         1 8 2 1
                        0013    803 ;            6.         1 8 1 F         1 8 0 0         1 8 1 F
                        0013    804 ;            7.         1 8 1 C         1 8 0 4         1 8 2 0
                        0013    805 ;            8.         1 8 1 B         1 8 0 3         1 8 1 E
                        0013    806 ;            9.         1 8 1 7         1 8 0 7         1 8 1 E
                        0013    807 ;            A.         1 8 1 D         1 8 0 9         1 8 2 6
                        0013    808 ;            B.         1 8 0 D         1 8 0 7         1 8 1 4
                        0013    809 ;            C.         1 8 1 D         1 8 0 1         1 8 1 E
                        0013    810 ;            D.         1 8 0 E         1 8 0 2         1 8 1 0
                        0013    811 ;
```

```
                 0013   ; LOGIC DESCRIPTION
                 0013   ;
                 0013   ;       This test examines the operation of a six bit adder in the MSQ gate
                 0013   ;       array. The outputs of the adder go through a multiplexer to bits
                 0013   ;       <05:00> of the control store address.
                 0013   ;
                 0013   ; ASSUMPTIONS
                 0013   ;
                 0013   ;       This test requires no signals from any other CPU modules.
                 0013   ;
                 0013   ; ERROR DESCRIPTION
                 0013   ;
                 0013   ;       On error this test will call out the MSQ gate array and the DPM module
                 0013   ;       name. The following information will also by typed out:
                 0013   ;
                 0013   ;       Expected Data:
                 0013   ;       Received Data:
                 0013   ;       Loop Count:
                 0013   ;
                 0013   ;--
                 0013   ;*******************************************************************************
                 0013
                 0013   ;///////////////////////////////////////////////////////////////////////////////
                 0013   ;+
                 0013   ; RDM pseudo-ops to implement this test.
                 0013   ;-
                 0013
                 0013           INITIALIZE                              ; INITIALIZE THE CPU
                 0015           LOOP            I,1,13                  ; LOOP TO EXERCISE MSQ ADDER
                 001C           LDFIELD         1$,I,WORD,6$,0,13       ; LOAD 1ST INPUT TO ADDER
                 002D           LDFIELD         2$,I,WORD,7$,0,13       ; LOAD 2ND INPUT TO ADDER
                 003E           LOADDCS         5$,,0,5                 ; LOAD DCS WITH TEST
                 0045           ERRLOOP                                 ; ERROR LOOP STARTS HERE
                 0047           FETCH           0                       ; EXECUTE THE ADDITION
                 004C           BRSTCLK         4                       ;       "       "
                 0050           CMPREGMSK       CSTRP,3$,I,4$,,WORD,    ; DOES CSTRP = SUM?
                 005C           IFERROR         ,<MSQ>,<DPM>            ; IF NOT SIGNAL AN ERROR
                 0063           ENDLOOP         I                       ; CONTINUE FOR ALL PATTERNS
                 0066           SKIP                                    ; TEST COMPLETED
                 006D
                 006D           BEGIN_TEST_DATA
                 006D
                 006D       ;-----------------------------------------------------------------------
                 006D
        1800     006D   1$:     .WORD   ^X 1800         ; Data Table 1
        183F     006F           .WORD   ^X 183F
        180A     0071           .WORD   ^X 180A
        1835     0073           .WORD   ^X 1835
        1803     0075           .WORD   ^X 1803
        181F     0077           .WORD   ^X 181F
        181C     0079           .WORD   ^X 181C
        181B     007B           .WORD   ^X 181B
        1817     007D           .WORD   ^X 1817
        181D     007F           .WORD   ^X 181D
        180D     0081           .WORD   ^X 180D
```

```
                        181D  0083   865          .WORD   ^X 181D
                        180E  0085   866          .WORD   ^X 180E
                              0087   867
                        183F  0087   868 2$:      .WORD   ^X 183F          ; Data Table 2
                        1801  0089   869          .WORD   ^X 1801
                        180A  008B   870          .WORD   ^X 180A
                        1835  008D   871          .WORD   ^X 1835
                        181E  008F   872          .WORD   ^X 181E
                        1800  0091   873          .WORD   ^X 1800
                        1804  0093   874          .WORD   ^X 1804
                        1803  0095   875          .WORD   ^X 1803
                        1807  0097   876          .WORD   ^X 1807
                        1809  0099   877          .WORD   ^X 1809
                        1807  009B   878          .WORD   ^X 1807
                        1801  009D   879          .WORD   ^X 1801
                        1802  009F   880          .WORD   ^X 1802
                              00A1   881
                        183F  00A1   882 3$:      .WORD   ^X 183F          ; Data Table 3 (adder sums)
                        1800  00A3   883          .WORD   ^X 1800
                        1814  00A5   884          .WORD   ^X 1814
                        182A  00A7   885          .WORD   ^X 182A
                        1821  00A9   886          .WORD   ^X 1821
                        181F  00AB   887          .WORD   ^X 181F
                        1820  00AD   888          .WORD   ^X 1820
                        181E  00AF   889          .WORD   ^X 181E
                        181E  00B1   890          .WORD   ^X 181E
                        1826  00B3   891          .WORD   ^X 1826
                        1814  00B5   892          .WORD   ^X 1814
                        181E  00B7   893          .WORD   ^X 181E
                        1810  00B9   894          .WORD   ^X 1810
                              00BB   895
                        003F  00BB   896 4$:      .WORD   ^X 003F          ; Mask for register comparison
                              00BD   897
                              00BD   898 5$:
U 00, 7700,C800,0364,0300,0470,1801   00BD   899 ;% 00    NOP                      ; Start of DCS
                              00C8   903 6$:
U 01, 7700,C800,0364,0300,0470,1802   00C8   904 ;% 01    NOP
U 02, 7700,C800,0364,0300,0470,5803   00D3   908 ;% 02    JSR/PUSH                 ; Push address onto micro-stack
                              00DE   912 7$:
U 03, 7700,8800,0364,0B00,0470,1804   00DE   913 ;% 03    BUT/RETURN               ; CSTRP = U-STACK + NEXT ADDRESS
U 04, 7300,4800,0364,0300,0470,1805   00E9   917 ;% 04    STOP.CLOCK               ; HALT
                              00F4   921
                              00F4   922 END_TEST_DATA
                              00F4
                              00F4       ;-------------------------------------------------------------------------
                              00F4   923
                              00F4   924 ENDTEST
```

```
0021          .SBTTL  "TEST   068     Check PCS RAM Data Integrity
0021    926 ;*****************************************************************************
0021    927 ;++
0021    928 ;
0021    929 ; FUNTIONAL DESCRIPTION:
0021    930 ;
0021    931 ;       This test will write six different data patterns to all of the PCS
0021    932 ;       locations and then read them back to insure there are no data integrity
0021    933 ;       problems.
0021    934 ;
0021    935 ; TEST ALGORITHM:
0021    936 ;
0021    937 ;        1.  Initialize the CPU
0021    938 ;        2.  Execute the microcode to:
0021    939 ;               a.  Disable Cache
0021    940 ;               b.  Enable the CMI
0021    941 ;        3.  Set error loop address
0021    942 ;        4.  Let RDM know that we expect a micro trap
0021    943 ;        5.  Execute the microcode to:
0021    944 ;               a.  Load VA with PCS Base address
0021    945 ;               b.  Write a test pattern to VA
0021    946 ;               c.  Give time for write to complete
0021    947 ;               d.  Read the location by the VA
0021    948 ;               e.  Pass read data back to RDM
0021    949 ;        6.  Check for micro trap.  If there is a micro trap, PCS (L0008) is
0021    950 ;            not installed on the system, then do the following:
0021    951 ;               a.  Execute the microcode to:
0021    952 ;                      i.  Point to Bus Error Register
0021    953 ;                     ii.  Pass the Bus Error Register back to the RDM
0021    954 ;               b.  Check to see if NXM bit is set.
0021    955 ;               c.  If NXM bit was not set then call an error, got a bit set
0021    956 ;                   that shouldn't have been.
0021    957 ;               d.  Skip to next test
0021    958 ;         7.  Check to see that read data matches the write pattern.
0021    959 ;         8.  Call an error if read data and write pattern did not match.
0021    960 ;         9.  Set-up a loop of 6 iterations
0021    961 ;        10.  Load the micro code to write a test pattern to all of PCS
0021    962 ;        11.  Load the WDREG with the test pattern
0021    963 ;        12.  Set-up DCS at starting micro address
0021    964 ;        13.  Insure that the Clocks are turned off
0021    965 ;        14.  Load match register with 1801(hex)
0021    966 ;        15.  Clear master halt enable
0021    967 ;        16.  Set bit to watch for micro traps
0021    968 ;        17.  Start the CPU clocks
0021    969 ;        18.  Perform a Watchdog loop while executing the microcode to:
0021    970 ;             (if loop count expires then force an error, shouldn't happen)
0021    971 ;               a.  Load RTEMP0 with the test pattern
0021    972 ;               b.  Load the VA with enable register address
0021    973 ;               c.  Write the enable register with 0 to enable writing of PCS
0021    974 ;               d.  Load the VA with PCS starting address
0021    975 ;               e.  Jump to DCS address 0
0021    976 ;               f.  Pass the current VA to the RDM
0021    977 ;               g.  Write the test pattern in RTEMP0
0021    978 ;               h.  Give time for the write to complete
0021    979 ;               i.  Update the VA by 4
```

```
0021   980 ;                    j.  Check to see if VA is equal to PCS last address + 4
0021   981 ;                          i.  If VA is equal to last address + 4 then there
0021   982 ;                              will be a micro match at address 1801
0021   983 ;                         ii.  If VA is Not equal to last address + 4 then
0021   984 ;                              microcode execution procedes as normal.
0021   985 ;                    k.  Jump to DCS address 0 ( goto step 18-f, continue
0021   986 ;                        writing the test pattern until VA is equal to PCS last
0021   987 ;                        address + 4)
0021   988 ;            19.  Load microcode to read all of PCS
0021   989 ;            20.  Set error loop address
0021   990 ;            21.  Set-up DCS at starting micro address
0021   991 ;            22.  Insure that the Clocks are turned off
0021   992 ;            23.  Load match register with 1801(hex)
0021   993 ;            24.  Clear master halt enable
0021   994 ;            25.  Set bit to watch for micro traps
0021   995 ;            26.  Start the CPU clocks
0021   996 ;            27.  Perform a Watchdog loop while executing the microcode to:
0021   997 ;                 (if loop count expires then force an error, shouldn't happen)
0021   998 ;                    a.  Load the VA with enable register address
0021   999 ;                    b.  Write a zero to VA which will enable writing/reading of
0021  1000 ;                        PCS
0021  1001 ;                    c.  Load the VA with PCS starting address
0021  1002 ;                    d.  Jump to DCS address 0
0021  1003 ;                    e.  Pass the current VA to the RDM
0021  1004 ;                    f.  Read the PCS location by the VA
0021  1005 ;                    g.  Mask out the low 20 bits (upper 12 bits no used)
0021  1006 ;                    h.  Check the read data to see if correct
0021  1007 ;                          i.  If not correct then there will be a micro match
0021  1008 ;                              at address 1801
0021  1009 ;                         ii.  If data correct then microcode execution procedes
0021  1010 ;                              as normal
0021  1011 ;                    i.  Check to see if VA is equal to PCS last address
0021  1012 ;                          i.  If VA = F0BFFC then there will be a micro match
0021  1013 ;                              at address 1801
0021  1014 ;                         ii.  If VA not equal to F0BFFC then microcode
0021  1015 ;                              execution procedes as normal
0021  1016 ;                    j.  Update the VA by 4
0021  1017 ;                    k.  Jump to DCS address 0 (goto step 27-e, continue reading
0021  1018 ;                        and checking data until either there is a data error or
0021  1019 ;                        until the last PCS address has been checked.
0021  1020 ;            28.  Check for a machine check
0021  1021 ;            29.  If there was a machine check then call an error
0021  1022 ;            30.  Check the WHREG (VA) to see that we made it to the last PCS
0021  1023 ;                 address.
0021  1024 ;            31.  If WHREG was not equal to PCS last address then there must have
0021  1025 ;                 been a data error, Call an error.
0021  1026 ;            32.  Continue with loop I for six test patterns ( goto step 10).
0021  1027 ;            33.  Skip to next test
0021  1028 ;
0021  1029 ; TEST PATTERNS:
0021  1030 ;
0021  1031 ;            The following six test patterns are used to check PCS RAM's:
0021  1032 ;
0021  1033 ;                    000AAAAA        Loop I = 1
0021  1034 ;                    00055555        Loop I = 2
```

```
0021  1035 ;                    00033333        Loop I = 3
0021  1036 ;                    000F0F0F        Loop I = 4
0021  1037 ;                    0000F0F0        Loop I = 5
0021  1038 ;                    000FFFFF        Loop I = 6
0021  1039 ;
0021  1040 ; LOGIC DESCRIPTION:
0021  1041 ;
0021  1042 ;       This test checks the data integrity of the RAM's on the PCS module.
0021  1043 ;       The address logic that feeds the RAM's is also checked.
0021  1044 ;
0021  1045 ; ASSUMPTIONS:
0021  1046 ;
0021  1047 ;       None
0021  1048 ;
0021  1049 ; ERROR DESCRIPTION:
0021  1050 ;
0021  1051 ;       First IFERROR:
0021  1052 ;
0021  1053 ;               Expected Read Data
0021  1054 ;               Received Read Data
0021  1055 ;
0021  1056 ;       Second IFERROR:
0021  1057 ;
0021  1058 ;               Contents of WHREG   <--\ These values are NOT significant as
0021  1059 ;               The Value 0F000000  <--/ used only to FORCE an error
0021  1060 ;               Loop I
0021  1061 ;               Loop J
0021  1062 ;
0021  1063 ;       Third IFERROR:
0021  1064 ;
0021  1065 ;               Contents of WHREG   <--\ These values are NOT significant as
0021  1066 ;               The Value 0F000000  <--/ used only to FORCE an error
0021  1067 ;               Loop I
0021  1068 ;               Loop J
0021  1069 ;
0021  1070 ;       Fourth IFERROR:
0021  1071 ;
0021  1072 ;               Contents of CSTRP register
0021  1073 ;               Machine Check Micro Trap Address
0021  1074 ;               Loop I
0021  1075 ;               Loop J
0021  1076 ;
0021  1077 ;       Fifth IFERROR:
0021  1078 ;
0021  1079 ;               Expected PCS Last Address
0021  1080 ;               Received PCS Last Address
0021  1081 ;               Loop I
0021  1082 ;               Loop J
0021  1083 ;               Expected Read Data
0021  1084 ;               Received Read Data
0021  1085 ;
0021  1086 ;       Sixth IFERROR:
0021  1087 ;
0021  1088 ;               Expected Bus Error Register
0021  1089 ;               Received Bus Error Register
```

```
0021  1090 ;
0021  1091 ;--
0021  1092 ;*****************************************************************************
0021  1093
0021  1094          INITIALIZE                      ; Initialize the CPU
0023  1095 ;
0023  1096 ; Disable Cache, Turn the CMI on, and disable CTRL^P interrupts
0023  1097 ;
0023  1098          FETCH           <^X19>          ; start dcs program
0028  1099          BRSTCLK         <^X20>,INH      ; end dcs program
002C  1100 ;
002C  1101 ; Check to see if there is a PCS on this system
002C  1102 ;
002C  1103          FETCH           <^X25>          ; start dcs program
0031  1104          BRSTCLK         <^X28>          ; end dcs program
0035  1105          ERRLOOP                         ; loop on error from here
0037  1106          EXPUTRAP                        ; Expect micro trap
0039  1107          FETCH           <^X11>          ; start dcs program
003E  1108          BRSTCLK         <^X18>          ; end dcs program
0042  1109          CMPREGMSK       CSTRP,7$,,8$,,W,NE
004E  1110                                          ; Was there a machine check
004E  1111          SKIP            150$,ONERROR    ; Go handle machine check
0055  1112          CMPREGMSK       WHREG,9$,,14$,,L; Check the Read Data
0061  1113          IFERROR         ,,<CCS>         ; Call error if data didn't match
0067  1114 ;
0067  1115 ; This routine will startup the micro code which will write the test pattern
0067  1116 ; each PCS location.
0067  1117 ;
0067  1118          LOOP            I,1,6           ; Set up loop for Six patterns
006E  1119          LOADDCS         70$,,0,<^X11>   ; Load the Read PCS microcode
0075  1120          LOADREG         WDREG,12$,I,L   ; Load the test pattern in WHREG
007D  1121          FETCH           <^X0A>          ; Set-up to start DCS
0082  1122          LOADREG         DCSCR,1$,,B     ; Insure that the clocks are off
008A  1123          LOADREG         CSAMR,2$,,W     ; Load the match address
0092  1124          LOADREG         RDCTRL,3$,,B    ; Clear master halt enable
009A  1125          LOADREG         RDCTRL,4$,,B    ; Watch for micro-traps
00A2  1126          LOADREG         DCSCR,5$,,B     ; Start the clocks
00AA  1127          LOOP            J,1,8192        ; Set up watch-dog loop counter
00B1  1128          CMPREGMSK       CLKDCS,3$,,6$,,B,
00BD  1129                                          ; Test to see if clock has stopped
00BD  1130          SKIP            100$,ONERROR    ; Exit loop if clock stops (normal)
00C4  1131          ENDLOOP         J               ; end of watch-dog loop
00C7  1132          CMPREG          WHREG,11$,,L,   ; Force an error, program shouldn't
00D0  1133          IFERROR         ,,<CCS>         ; get here from the loop
00D6  1134
00D6  1135 ;
00D6  1136 ; This routine will startup the micro code which will read each PCS location
00D6  1137 ; and test it to make sure it is valid.
00D6  1138 ;
00D6  1139
00D6  1140 100$:    LOADDCS         60$,,0,<^X11>   ; Load the Read PCS microcode
00DD  1141          ERRLOOP                         ; loop on error from here
00DF  1142          FETCH           <^X0C>          ; Set-up to start DCS
00E4  1143          LOADREG         DCSCR,1$,,B     ; Insure that the clocks are off
00EC  1144          LOADREG         CSAMR,2$,,W     ; Load the match address
```

```
                00F4  1145          LOADREG         RDCTRL,3$,,B    ; Clear master halt enable
                00FC  1146          LOADREG         RDCTRL,4$,,B    ; Watch for micro-traps
                0104  1147          LOADREG         DCSCR,5$,,B     ; Start the clocks
                010C  1148          LOOP            J,1,8192        ; Set up watch-dog loop counter
                0113  1149          CMPREGMSK       CLKDCS,3$,,6$,,B,
                011F  1150                                          ; Test to see if clock has stopped
                011F  1151          SKIP            110$,ONERROR    ; Exit loop if clock stops (normal)
                0126  1152          ENDLOOP         J               ; end of watch-dog loop
                0129  1153          CMPREG          WHREG,11$,,L,   ; Force an error, program shouldn't
                0132  1154          IFERROR         ,,<CCS>         ; get here from the loop
                0138  1155          SKIP
                013F  1156 110$:
                013F  1157          CMPREGMSK       CSTRP,7$,,8$,,W,NE
                014B  1158                                          ; Was there a machine check
                014B  1159          IFERROR         ,,<CCS>         ; Yes, call an error
                0151  1160          CMPREGMSK       WHREG,13$,,15$,,L,
                015D  1161                                          ; Check the address to see if finished
                015D  1162          IFERROR         2,,<CCS>        ; Data is Not correct, call error
                0163  1163          ENDLOOP         I               ; Restart for next pattern
                0166  1164          SKIP                            ; Goto next test
                016D  1165 ;
                016D  1166 ; This is the machine check handler routine.  It makes sure that the Non
                016D  1167 ; Existent memory bit is set which should be the case.
                016D  1168 ;
                016D  1169 150$:    FETCH           <^X21>          ; Start DCS program
                0172  1170          BRSTCLK         <^X24>,INH      ; End DCS program
                0176  1171          CMPREGMSK       WHREG,10$,,11$,,L
                0182  1172                                          ; Check to see if Non-Existent memory
                0182  1173          IFERROR         ,,<CCS>         ; A Bit is set that shouldn't be
                0188  1174          SKIP
                018F  1175
                018F  1176
                018F  1177 BEGIN_TEST_DATA
                018F
                018F        ;-------------------------------------------------------------------------
                018F  1178
          60    018F  1179 1$:      .BYTE   ^X60                    ; Data to shut the clocks off
        1801    0190  1180 2$:      .WORD   ^X1801                  ; Match Address
          00    0192  1181 3$:      .BYTE   ^X00                    ; Data to clear master halt enable
          28    0193  1182 4$:      .BYTE   ^X28                    ; Data to watch for u-traps
          11    0194  1183 5$:      .BYTE   ^X11                    ; Run clock bits
          C0    0195  1184 6$:      .BYTE   ^XC0                    ; Mask for clock stop bits
        0028    0196  1185 7$:      .WORD   ^X0028                  ; Machine Check miro trap
        3FFF    0198  1186 8$:      .WORD   ^X3FFF                  ; Mask for cmpregmsk pseudo op
    000AAAAA    019A  1187 9$:      .LONG   ^X000AAAAA              ; Expected data if PCS present
    0A000000    019E  1188 10$:     .LONG   ^X0A000000              ; Expected Bus Error bits
    0F000000    01A2  1189 11$:     .LONG   ^X0F000000              ; Mask for Bus Error Reg bits
                01A6  1190
    C00AAAAA    01A6  1191 12$:     .LONG   ^X000AAAAA              ; These are the test patterns that will
    00055555    01AA  1192          .LONG   ^X00055555              ; will be wrote to all of PCS RAM.
    00033333    01AE  1193          .LONG   ^X00033333              ; Note that Bit 20, Patch Present, is
    000F0F0F    01B2  1194          .LONG   ^X000F0F0F              ; not asserted in these patterns.
    0000F0F0    01B6  1195          .LONG   ^X0000F0F0              ; ...
    000FFFFF    01BA  1196          .LONG   ^X000FFFFF              ; ...
                01BE  1197
```

```
                    00F0C000  01BE  1198 13$:    .LONG   ^X00F0C000              ; PCS last address + 4
                              01C2  1199
                    000FFFFF  01C2  1200 14$:    .LONG   ^X000FFFFF              ; Mask for PCS data compare
                              01C6  1201
                    00FFFFFF  01C6  1202 15$:    .LONG   ^X00FFFFFF              ; Mask for address compare
                              01CA  1203
                              01CA  1204 ;
                              01CA  1205 ; The following routine will read all PCS locations and check for the correct
                              01CA  1206 ; test pattern.
                              01CA  1207 ;
                              01CA  1208
                              01CA  1209 60$:
U 00, 7700,C800,0364,0300,0470,1802   01CA  1210 ;* 00   NOP,NEXT/1802                ; Get the CPU up to speed
U 01, 7700,C800,0364,0300,0470,1802   01D5  1214 ;* 01   NOP                         ; ...
U 02, 6700,081B,5924,0300,0470,1803   01E0  1218 ;* 02   RDM_WB,WB_M[VA]             ; Pass the VA back to the RDM
U 03, 7700,4800,0364,0300,0400,1804   01EB  1222 ;* 03   READ.PHY                    ; Read the PCS by the VA
U 04, 7700,3800,7FF8,0000,0470,1805   01F6  1226 ;* 04   LONLIT_[000FFFFF]          ; Load lonlit register with mask value
U 05, 7700,0812,0022,03D4,0470,1806   0201  1230 ;* 05   D_M[MDR].AND.R[LONLIT]     ; Mask out upper 12 bits (unused)
U 06, 7700,8861,5B24,0300,0470,1807   020C  1234 ;* 06   M[TEMP1]_D                 ; Load M[TEMP1] with masked data
U 07, 7700,0801,0034,A300,0470,1800   0217  1238 ;* 07   WB_M[TEMP1].XOR.R[TEMP0],BUT/WX.EQ.0,NEXT/1800
                              0222  1242                                          ; Check the read data to see if correct
U 08, 7700,4800,0364,0300,0470,1809   0222  1243 ;* 08   NOP                         ; ...
U 09, 7700,4800,0364,0300,0470,180A   022D  1247 ;* 09   NOP                         ; ...
U 0A, 7700,481B,0034,A30C,0470,1800   0238  1251 ;* 0A   WB_M[VA].XOR.R[TEMP3],BUT/WX.EQ.0,NEXT/1800
                              0243  1255                                          ; check to see if finished
U 0B, 7700,C800,0364,0300,4470,180C   0243  1256 ;* 0B   VA_VA+4                    ; Update the VA for next patch bit
U 0A, 7500,C800,0364,0300,0470,180B   024E  1260 ;* 0A   NOP,DCS.ADDR_0            ; Continue with patch bit initialize
U 0C, 7700,C800,0364,0300,0470,180D   0259  1264 ;* 0C   NOP
U 0D, 7700,8800,5BE4,031C,4A70,180E   0264  1268 ;* 0D   VA_R[TEMP7]               ; Load VA with register address to
                              026F  1272                                          ; enable the reading of PCS
U 0E, 7700,4800,5BE4,03D8,5C80,980F   026F  1273 ;* 0E   WRITE.PHY R[ZERO],CLOCK.EXT ; Enable PCS
U 0F, 7700,8800,5BE4,0310,4A70,1810   027A  1277 ;* 0F   VA_R[TEMP4]               ; Load VA with PCS Base address
U 10, 7500,4800,0364,0300,0470,1811   0285  1281 ;* 10   NOP,DCS.ADDR_0            ; Go write the PCS
                              0290  1285
                              0290  1286 ;
                              0290  1287 ; The following routine will write the test pattern to all PCS locations
                              0290  1288 ;
                              0290  1289
                              0290  1290 70$:
U 00, 7700,C800,0364,0300,0470,1802   0290  1291 ;* 00   NOP,NEXT/1802                ; Get the CPU up to speed
U 01, 7700,C800,0364,0300,0470,1802   029B  1295 ;* 01   NOP                         ; ...
U 02, 6700,881B,5924,0300,0470,9803   02A6  1299 ;* 02   RDM_M[VA],CLOCK.EXT        ; Save the VA in the RDM
U 03, 7700,4800,5BE4,0300,5C80,1804   02B1  1303 ;* 03   WRITE.PHY R[TEMP0]         ; Write the PCS location by VA
U 04, 7700,C800,0364,0300,0470,9805   02BC  1307 ;* 04   NOP,CLOCK.EXT             ; Wait for write to finish
U 05, 7700,C800,0364,0300,0470,9806   02C7  1311 ;* 05   NOP,CLOCK.EXT             ; ...
U 06, 7700,4800,0364,0300,4470,1807   02D2  1315 ;* 06   VA_VA+4                    ; Update the VA for next patch bit
U 07, 7700,081B,0034,A31C,0470,1800   02DD  1319 ;* 07   WB_M[VA].XOR.R[TEMP7],BUT/WX.EQ.0,NEXT/1800
                              02E8  1323                                          ; check to see if finished
U 08, 7700,4800,0364,0300,0470,1809   02E8  1324 ;* 08   NOP                         ; ...
U 09, 7500,4800,0364,0300,0470,180A   02F3  1328 ;* 09   NOP,DCS.ADDR_0            ; Continue with write of PCS
U 0A, 7700,C800,0364,0300,0470,180B   02FE  1332 ;* 0A   NOP
U 0B, 5700,0840,0364,0300,0470,180C   0309  1336 ;* 0B   R[TEMP0]_RDM               ; Put test pattern in R[TEMP8]
U 0C, 7700,8800,5BE4,031C,4A70,180D   0314  1340 ;* 0C   VA_R[TEMP7]               ; Load VA with register address to
                              031F  1344                                          ; enable the writing of PCS
U 0D, 7700,4800,5BE4,03D8,5C80,180E   031F  1345 ;* 0D   WRITE.PHY R[ZERO]         ; Enable to write enable register
```

```
U 0E. 7700,C800,0364,0300,0470,980F  032A  1349 ;% 0E   NOP,CLOCK.EXT                    ; Wait for write to finish
U 0F. 7700,8800,5BE4,0310,4A70,1810  0335  1353 ;% 0F   VA_R[TEMP4]                     ; Load VA with PCS Base Address
U 10. 7500,4800,0364,0300,0470,1811  0340  1357 ;% 10   NOP,DCS.ADDR_0                 ; Go write the PCS
                                     034B  1361
                                     034B  1362 BEGIN_CPU_DATA
                                     0002  1363
                                     0002  1364 DCS_DATA
                                     0001  1365
                                     0001  1366 ;
                                     0001  1367 ; The following routine will write the test pattern to all PCS locations
                                     0001  1368 ;
                                     0001  1369
U 00. 7700,C800,0364,0300,0470,1802  0001  1370 ;% 00   NOP,NEXT/1802                  ; Get the CPU up to speed
U 01. 7700,C800,0364,0300,0470,1802  000C  1374 ;% 01   NOP                           ; ...
U 02. 6700,881B,5924,0300,0470,9803  0017  1378 ;% 02   RDM_M[VA],CLOCK.EXT           ; Save the VA in the RDM
U 03. 7700,4800,5BE4,0300,5C80,1804  0022  1382 ;% 03   WRITE.PHY R[TEMP0]            ; Write the PCS location by VA
U 04. 7700,C800,0364,0300,0470,9805  002D  1386 ;% 04   NOP,CLOCK.EXT                 ; Wait for write to finish
U 05. 7700,C800,0364,0300,0470,9806  0038  1390 ;% 05   NOP,CLOCK.EXT                 ; ...
U 06. 7700,4800,0364,0300,4470,1807  0043  1394 ;% 06   VA_VA+4                       ; Update the VA for next patch bit
U 07. 7700,081B,0034,A31C,0470,1800  004E  1398 ;% 07   WB_M[VA].XOR.R[TEMP7],BUT/WX.EQ.0,NEXT/1800
                                     0059  1402                                     ; check to see if finished
U 08. 7700,4800,0364,0300,0470,1809  0059  1403 ;% 08   NOP                           ; ...
U 09. 7500,4800,0364,0300,0470,180A  0064  1407 ;% 09   NOP,DCS.ADDR_0                ; Continue with write of PCS
U 0A. 7700,C800,0364,0300,0470,180B  006F  1411 ;% 0A   NOP
U 0B. 5700,0840,0364,0300,0470,180C  007A  1415 ;% 0B   R[TEMP0]_RDM                  ; Put test pattern in R[TEMP8]
U 0C. 7700,8800,5BE4,031C,4A70,180D  0085  1419 ;% 0C   VA_R[TEMP7]                   ; Load VA with register address to
                                     0090  1423                                     ; enable the writing of PCS
U 0D. 7700,4800,5BE4,03D8,5C80,180E  0090  1424 ;% 0D   WRITE.PHY R[ZERO]            ; Enable to write enable register
U 0E. 7700,C800,0364,0300,0470,980F  009B  1428 ;% 0E   NOP,CLOCK.EXT                 ; Wait for write to finish
U 0F. 7700,8800,5BE4,0310,4A70,1810  00A6  1432 ;% 0F   VA_R[TEMP4]                   ; Load VA with PCS Base Address
U 10. 7500,4800,0364,0300,0470,1811  00B1  1436 ;% 10   NOP,DCS.ADDR_0               ; Go write the PCS
                                     00BC  1440
                                     00BC  1441 ;
                                     00BC  1442 ; The following routine will first write a test pattern to the first PCS
                                     00BC  1443 ; then read the test data.  The data will be sent back to the RDM and if no
                                     00BC  1444 ; PCS is present then the data will not be valid.
                                     00BC  1445 ;
                                     00BC  1446
U 11. 7700,4800,0364,0300,0470,1812  00BC  1447 ;% 11   NOP                           ; Get the CPU up to speed
U 12. 7700,8800,5BE4,0310,4A70,1813  00C7  1451 ;% 12   VA_R[TEMP4]                   ; Load VA with PCS Base Address
U 13. 7700,C800,5BE4,0318,5C80,1814  00D2  1455 ;% 13   WRITE.PHY R[TEMP6]           ; Write the Test pattern
U 14. 7700,4800,0364,0300,0470,9815  00DD  1459 ;% 14   NOP,CLOCK.EXT                 ; Give write time to complete
U 15. 7700,4800,0364,0300,0470,9816  00E8  1463 ;% 15   NOP,CLOCK.EXT                 ; ...
U 16. 7700,4800,0364,0300,0400,9817  00F3  1467 ;% 16   READ.PHY,CLOCK.EXT            ; Now Read the Test location
U 17. 6700,0812,5924,0300,0470,1818  00FE  1471 ;% 17   RDM_WB,WB_M[MDR]             ; Pass the read data back to the RDM
U 18. 7300,C800,0364,0300,0470,1819  0109  1475 ;% 18   NOP,STOP.CLOCK              ; Return to the RDM
                                     0114  1479
                                     0114  1480 ;
                                     0114  1481 ; The following routine will Disable Cache and Enable the CMI
                                     0114  1482 ;
                                     0114  1483
U 19. 7700,C800,0364,0300,0470,181A  0114  1484 ;% 19   NOP                           ; Get CPU up to speed
U 1A. 7700,1800,CB70,0307,6870,181B  011F  1488 ;% 1A   MEMSCAR_ZLIT24[0E]          ; Point MEMSCAR to CMI Disable reg
U 1B. 7700,9800,CB70,0300,6070,181C  012A  1492 ;% 1B   MEMSCR_ZLIT24[00]           ; Enable the CMI
U 1C. 7700,5800,CB70,0303,6870,181D  0135  1496 ;% 1C   MEMSCAR_ZLIT24[06]          ; Point MEMSCAR to Cache Disable reg
```

```
U 1D, 7700,5800,CB70,0300,E070,181E  0140  1500 ;% 1D   MEMSCR_ZLIT24[01]                ; Disable Cache
U 1E, 7700,5800,D370,0306,2470,181F  014B  1504 ;% 1E   CRAR_ZLIT16[0C]                 ; Point CRAR to Command & Status reg.
U 1F, 7700,D800,D370,0303,2C70,1820  0156  1508 ;% 1F   CONREGS_ZLIT16[06]             ; Disable Ctrl P interrupts
U 20, 7300,4800,0364,0300,0470,1821  0161  1512 ;% 20   NOP,STOP.CLOCK                 ; Return to the RDM
                                     016C  1516
                                     016C  1517 ;
                                     016C  1518 ; The following routine will read the Bus Error Register.
                                     016C  1519 ;
                                     016C  1520
U 21, 7700,4800,0364,0300,0470,1822  016C  1521 ;% 21   NOP                           ; Get CPU up to speed
U 22, 7700,1800,CB70,0304,E870,1823  0177  1525 ;% 22   MEMSCAR_ZLIT24[09]            ; Point MEMSCAR to Bus Error Register
U 23, 6700,4800,0364,0300,6470,1824  0182  1529 ;% 23   RDM_WB,WCTRL/MEMSCR           ; Read Bus Error Register
U 24, 7300,C800,0364,0300,0470,1825  018D  1533 ;% 24   NOP,STOP.CLOCK               ; Return to the RDM
                                     0198  1537
                                     0198  1538 ;
                                     0198  1539 ; This routine will enable writing of PCS/WCS
                                     0198  1540 ;
                                     0198  1541
U 25, 7700,C800,0364,0300,0470,1826  0198  1542 ;% 25   NOP                           ; Get CPU up to speed
U 26, 7700,0800,5BE4,031C,4A70,1827  01A3  1546 ;% 26   VA_R[TEMP7]                   ; Load VA with enable register address
U 27, 7700,C800,5BE4,03D8,5C80,1828  01AE  1550 ;% 27   WRITE.PHY R[ZERO]            ; Write zero (Enable writing of PCS)
U 28, 7300,C800,0364,0300,0470,1829  01B9  1554 ;% 28   NOP,STOP.CLOCK               ; Return to the RDM
                                     01C4  1558
                                     01C4  1559 RTEMP_DATA
                                     0001  1560
                          00000000   0001  1561         .LONG   ^X00000000          ; R[TEMP0] Expected Data
                          00000000   0005  1562         .LONG   ^X00000000          ; R[TEMP1] Received Data
                          00000000   0009  1563         .LONG   ^X00000000          ; R[TEMP2]
                          00F0BFFC   000D  1564         .LONG   ^X00F0BFFC          ; R[TEMP3]
                          00F08000   0011  1565         .LONG   ^X00F08000          ; Starting Address of PCS
                          00F00000   0015  1566         .LONG   ^X00F00000          ; Starting Address for Patch Bits
                          000AAAAA   0019  1567         .LONG   ^X000AAAAA          ; Test pattern used to check for PCS
                                     001D  1568                                     ; present - Note that Bit 20 is 0 which
                                     001D  1569                                     ; will force PCS Present to be Not True
                          00F0C000   001D  1570         .LONG   ^X00F0C000          ; WCS/PCS or Patch Bit write enable reg
                          00000000   0021  1571         .LONG   ^X00000000          ; Storage for current test pattern
                                     0025  1572
                                     0025  1573 END_CPU_DATA
                                     034B  1574
                                     034B  1575 END_TEST_DATA
                                     034B
                                     034B       ;-----------------------------------------------------------------------
                                     034B  1576
                                     034B  1577 ENDTEST
```

```
                        .SBTTL  "TEST   069     Control store parity error micro-trap test
                1579 ;*******************************************************************************
                1580 ;++
                1581 ;
                1582 ; FUNCTIONAL DESCRIPTION
                1583 ;
                1584 ;         This test will examine the control store parity error logic for pro-
                1585 ;       per operation. This test is comprised of two subtests.The first subtest
                1586 ;       examines the logic that implements the CS parity error micro-trap,while
                1587 ;       the second subtest examines the same micro - trap  during a  BUT/RETURN
                1588 ;       operation.
                1589 ;
                1590 ; TEST ALGORITHM
                1591 ;
                1592 ;       Subtest 1
                1593 ;
                1594 ;       Examine the control store parity logic for a CS address of 20 hex
                1595 ;       during a control store parity error.
                1596 ;
                1597 ;       1.  Initialize the CPU.
                1598 ;       2.  Change the parity field of the micro-instruction for the first
                1599 ;           subtest to force a control store parity error.
                1600 ;       3.  Load the micro-instructions for the first subtest into DCS.
                1601 ;       4.  Prepare the RDM to expect a control store parity error.
                1602 ;       5.  Execute the micro-instructions:
                1603 ;               1. NOP
                1604 ;               2. NEXT/3FFF,STOP.CLOCK ; Micro-instruction with CSPE
                1605 ;       6.  Compare the contents of the CS address trace register with the
                1606 ;           expected control store address of 20 hex.
                1607 ;       7.  If the comparison shows that the two are not equal, signal an
                1608 ;           error.
                1609 ;
                1610 ;       Subtest 2
                1611 ;
                1612 ;       Examine the control store parity logic for a CS address of 20 hex
                1613 ;       when the CPU is executing a BUT/RETURN operation.
                1614 ;
                1615 ;       8.  Change the parity field of the micro-instruction for the second
                1616 ;           subtest to force a control store parity error.
                1617 ;       9.  Load the micro-instructions for the second subtest into DCS.
                1618 ;       10. Prepare the RDM to expect a control store parity error.
                1619 ;       11. Execute the micro-instructions:
                1620 ;               3. NOP
                1621 ;               4. JSR/PUSH,NEXT/3FFF
                1622 ;               5. BUT/RETURN,NEXT/0000,STOP.CLOCK      ; CSPE here
                1623 ;       12. Compare the contrents of the CS address trace register with the
                1624 ;           expected control store address of 20 hex.
                1625 ;       13. If the comparison shows that the two are not equal, signal an
                1626 ;           error.
                1627 ;       14. Test completed.
                1628 ;
                1629 ; DATA PATTERNS
                1630 ;
                1631 ;       There are three data pattern used in this test:
                1632 ;
```

```
                        002F  1633 ;       1. 3FFF hex fills the next address field with ones.
                        002F  1634 ;
                        002F  1635 ;       2. 0020 hex is the expected micro-address for a control store parity
                        002F  1636 ;          error.
                        002F  1637 ;
                        002F  1638 ;       3. 0000 hex is OR-ed with 3FFF in the BUT/RETURN operation for the
                        002F  1639 ;          second subtest.
                        002F  1640 ;
                        002F  1641 ; LOGIC DESCRIPTION
                        002F  1642 ;
                        002F  1643 ;       This test examines the MSQ gate array for the proper generation of the
                        002F  1644 ;       signal ZERO HIGH NEXT on DPM-14. ZERO HIGH NEXT runs to DPM-13 and
                        002F  1645 ;       provides the enable to the tri-state drivers on CS bits <13:06>. These
                        002F  1646 ;       drivers force a zero onto bits <13:06> of the CS address whenever ZERO
                        002F  1647 ;       HIGH NEXT is active.
                        002F  1648 ;
                        002F  1649 ;       Also tested is the micro-vector logic which is hard wired onto bits
                        002F  1650 ;       <05:00> of the CS address with the MSQ gate array using open collector
                        002F  1651 ;       logic. The micro-vector enable signal comes from the SAC gate array.
                        002F  1652 ;
                        002F  1653 ; ASSUMPTIONS
                        002F  1654 ;
                        002F  1655 ;       This test requires signals from the UTR and ACV gate arrays on the
                        002F  1656 ;       MIC module.
                        002F  1657 ;
                        002F  1658 ; ERROR DESCRIPTION
                        002F  1659 ;
                        002F  1660 ;       On error this test will call out the MSQ and SAC gate arrays on the
                        002F  1661 ;       DPM module and the UTR and ACV gate arrays on the MIC module.
                        002F  1662 ;
                        002F  1663 ;       The following information is also typed out:
                        002F  1664 ;
                        002F  1665 ;       Expected CS Address
                        002F  1666 ;       Received CS Address
                        002F  1667 ;
                        002F  1668 ;--
                        002F  1669 ;*******************************************************************************
                        002F  1670
                        002F  1671          SUBTEST         1
                        0032
                        0032       ;///////////////////////////////////////////////////////////////////////////
                        0032  1672 ;+
                        0032  1673 ; Subtest 1
                        0032  1674 ;
                        0032  1675 ;       This subtest examines the CS parity error logic to make sure that it
                        0032  1676 ;       generates an address of 20 hex whenever a CS parity error occurrs.
                        0032  1677 ;-
                        0032  1678          LDFIELD         3$,,Byte,11$,78,78,,,,NP; Load Micro word
                        0043  1679          LOADDCS         10$,,0,3                ; LOAD TEST U-CODE
                        004A  1680          EXPUTRAP                                ; EXPECT A CSPE U-TRAP
                        004C  1681          ERRLOOP                                 ; ERROR LOOP HERE
                        004E  1682          INITIALIZE                              ; INITIALIZE THE CPU
                        0050  1683          FETCH           0                       ; EXECUTE CODE WITH BAD PARITY
                        0055  1684          BRSTCLK         2,INH                   ; NO DCS LOOPING ON ERROR
                        0059  1685          CMPREGMSK       CSTRP,1$,,2$,,WORD      ; DOES CS ADDRESS = CPSE?
```

```
                        0065  1686          IFERROR         ,<<MSQ>,<SAC>,<ACV>-    ; IF NOT ERROR CALL
                        0065  1687                          ,<UTR>>,<MIC>           ; ...
                        006F  1688
                        006F  1689          SUBTEST         2
                        0072
                        0072        ;////////////////////////////////////////////////////////////////////////////
                        0072  1690  ;+
                        0072  1691  ; Subtest 2
                        0072  1692  ;
                        0072  1693  ;       This subtest examines the CS parity error logic when the CPU is
                        0072  1694  ;       executing a BUT/RETURN operation. The address that should appear on
                        0072  1695  ;       the CS lines is 20 hex, regardless of the return address.
                        0072  1696  ;-
                        0072  1697          LDFIELD         3$,,Byte,21$,78,78,,,,NP; Load Micro word
                        0083  1698          LOADDCS         20$,,0,4                ; LOAD BAD U-INSTRUCTION
                        008A  1699          EXPUTRAP                                ; EXPECT A CSPE U-TRAP
                        008C  1700          ERRLOOP                                 ; ERROR LOOP HERE
                        008E  1701          INITIALIZE                              ; INITIALIZE THE CPU
                        0090  1702          FETCH           0                       ; EXECUTE THE BUT/RETURN
                        0095  1703          BRSTCLK         3,INH                   ; NO DCS LOOP ON ERROR
                        0099  1704          CMPREGMSK       CSTRP,1$,,2$,,WORD      ; DOES CS ADDRESS = CSPE?
                        00A5  1705          IFERROR         ,<<MSQ><SAC>>,          ; IF NOT SIGNAL AN ERROR
                        00AC  1706          SKIP                                    ; END OF TEST
                        00B3  1707
                        00B3  1708          BEGIN_TEST_DATA
                        00B3
                        00B3        ;-------------------------------------------------------------------------
                        00B3  1709
                  0020  00B3  1710  1$:     .WORD   ^X 0020         ; CS ADDRESS EXPECTED ON A CSPE MICRO-TRAP
                  1FFF  00B5  1711  2$:     .WORD   ^X 1FFF         ; MASK FOR REGISTER COMPARISON
                    00  00B7  1712  3$:     .Byte   0               ; Clear parity bit
                        00B8  1713
                        00B8  1714  10$:
U 00, 7700,C800,0364,0300,0470,1801     00B8  1715  ;% 00   NOP                     ; START OF DCS
                        00C3  1719  11$:
U 01, 7700,C800,0364,0300,0470,1802     00C3  1720  ;% 01   NOP                     ; FORCE BAD PARITY WITH LDFIELD OP
U 02, 7300,4800,0364,0300,0470,1803     00CE  1724  ;% 02   STOP.CLOCK              ; HALT
                        00D9  1728
                        00D9  1729  20$:
U 00, 7700,C800,0364,0300,0470,1801     00D9  1730  ;% 00   NOP                     ; PAUSE TO GET CPU UP TO SPEED
U 01, 7700,4800,0364,0300,0470,5802     00E4  1734  ;% 01   JSR/PUSH                ; PUSH AN ADDRESS ONTO THE MICRO STACK
                        00EF  1738                                                  ; EXECUTE A RETURN FUNCTION WITH THE RSRC FIELD
                        00EF  1739                                                  ; KLUGED TO FORCE A PARITY ERROR
                        00EF  1740  21$:
U 02, 7700,4800,0364,0B04,0470,0000     00EF  1741  ;% 02   RSRC/TEMP1,BUT/RETURN,NEXT/0000 ; FORCE PARITY ERROR WITH LDFIELD OP
U 03, 7300,C800,0364,0300,0470,1804     00FA  1745  ;% 03   STOP.CLOCK              ; HALT
                        0105  1749
                        0105  1750  END_TEST_DATA
                        0105
                        0105        ;-------------------------------------------------------------------------
                        0105  1751
                        0105  1752  ENDTEST
```

```
                        001F        .SBTTL  "TEST   06A     TEST XB0 AND XB1 REGISTERS
                        001F  1754 ;*******************************************************************************
                        001F  1755 ;++
                        001F  1756 ;
                        001F  1757 ; FUNCTIONAL DESCRIPTION:
                        001F  1758 ;
                        001F  1759 ;       THIS TEST CONSISTS OF 2 SUBTESTS TO CHECK THE DATA INTEGRITY OF THE
                        001F  1760 ;       EXECUTION BUFFERS.  6 TEST PATTERNS ARE WRITTEN TO AND READ BACK
                        001F  1761 ;       FROM EACH REGISTER.
                        001F  1762 ;
                        001F  1763 ;
                        001F  1764 ; TEST ALGORITHM:
                        001F  1765 ;
                        001F  1766 ;       1.  CREATE A TEST LOOP
                        001F  1767 ;       2.  LOAD DCS WITH MICRO WORDS FOR THIS TEST
                        001F  1768 ;       3.  EXECUTE DCS PROGRAM
                        001F  1769 ;               1.  LOAD ADDRESS OF PHYSICAL/VIRTUAL ADDRESSING REGISTER INTO
                        001F  1770 ;                   MEMORY STATUS AND CONTROL ADDRESS REGISTER (MEMSCAR)
                        001F  1771 ;               2.  SET MEMORY MANAGEMENT OFF
                        001F  1772 ;               3.  LOAD ADDRESS OF REFERENCE CACHE ONLY REGISTER INTO MEMSCAR
                        001F  1773 ;               4.  SET CMI OFF
                        001F  1774 ;               5.  LOAD VIRTUAL ADDRESS REGISTER WITH 0'S
                        001F  1775 ;               6.  PUT TEST PATTERN INTO LONGLIT REGISTER
                        001F  1776 ;               7.  WRITE TEST PATTERN IN LONGLIT TO CACHE ADDRESS 0
                        001F  1777 ;               8.  INCREMENT VITUAL ADDRESS REGISTER BY 4
                        001F  1778 ;               9.  PUT TEST PATTERN INTO LONGLIT REGISTER
                        001F  1779 ;              10.  WRITE TEST PATTERN IN LONGLIT TO CACHE ADDRESS 4
                        001F  1780 ;              11.  LOAD PROGRAM COUNTER WITH 0'S
                        001F  1781 ;              12.  READ DESIRED TEST PATTERN FROM XB'S TO RDM
                        001F  1782 ;       4.  COMPARE RESULTS (EXPECTED AND RECEIVED)
                        001F  1783 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST ITERATIONS
                        001F  1784 ;
                        001F  1785 ;
                        001F  1786 ; TEST PATTERNS:
                        001F  1787 ;
                        001F  1788 ;       ITERATION           XB0             XB1
                        001F  1789 ;
                        001F  1790 ;           1           AAAAAAAA        0000FFFF
                        001F  1791 ;           2           55555555        00FF00FF
                        001F  1792 ;           3           33333333        0F0F0F0F
                        001F  1793 ;           4           0F0F0F0F        33333333
                        001F  1794 ;           5           00FF00FF        55555555
                        001F  1795 ;           6           0000FFFF        AAAAAAAA
                        001F  1796 ;
                        001F  1797 ;
                        001F  1798 ; LOGIC DESCRIPTION:
                        001F  1799 ;
                        001F  1800 ;       BY DIRECTLY LOADING THE PC THIS TEST INVOKES THE PREFETCH MECHANISM
                        001F  1801 ;       TO LOAD THE XB'S.  IN THE EVENT OF ERRORS THE "MDR" GATE ARRAY IS A
                        001F  1802 ;       GOOD FIRST CHOICE TO SWAP.  IT CONTAINS MOST OF THE DATA PATH
                        001F  1803 ;       INVOLVED IN THIS TEST.  SECOND CHOICE SHOULD BE "ADD" WHICH IS
                        001F  1804 ;       RESPONSIBLE FOR THE ADDRESSING.  IF PROBLEMS STILL PERSIST, THEN YOU
                        001F  1805 ;       MIGHT TRY SWAPPING "PRK" WHICH CONTROLS THE PREFETCH LOGIC.
                        001F  1806 ;       TO HELP ISOLATE PROBLEMS THE FIRST SUBTEST EXAMINES THE MSRC XB H
                        001F  1807 ;       SIGNAL ON THE VBUS FOR YOU.
```

```
                        001F  1808 ;       IF XB1 CONSISTANTLY CONTAINS XB0 DATA YOU MIGHT CHECK THE LATCH MA
                        001F  1809 ;       SIGNAL COMING FROM THE PRK GATE ARRAY.  THIS SIGNAL CAN MESS-UP THE
                        001F  1810 ;       PREFETCH LOGIC WHICH IS INVOKED WHEN THE PC IS LOADED.
                        001F  1811 ;
                        001F  1812 ;
                        001F  1813 ; ASSUMPTIONS:
                        001F  1814 ;
                        001F  1815 ;       THIS TEST ASSUMES THE VA, PC, CACHE, WBUS, AND MBUS ARE OPERATIONAL.
                        001F  1816 ;
                        001F  1817 ;
                        001F  1818 ; ERROR DESCRIPTION:
                        001F  1819 ;
                        001F  1820 ;       SUBTEST #1 FIRST IFERROR:
                        001F  1821 ;              EXPECTED VBUS DATA (MSRC XB L)
                        001F  1822 ;              RECEIVED VBUS DATA
                        001F  1823 ;              LOOP COUNT I
                        001F  1824 ;                     (BITS <7:0> = BIT POSITION)
                        001F  1825 ;                     (BITS <8> = BIT STATE)
                        001F  1826 ;
                        001F  1827 ;       SUBTEST #1 SECOND IFERROR AND ONLY IFERROR IN ALL OTHERS:
                        001F  1828 ;              EXPECTED XB DATA
                        001F  1829 ;              RECEIVED XB DATA
                        001F  1830 ;              LOOP COUNT I
                        001F  1831 ;
                        001F  1832 ;--
                        001F  1833 ;*****************************************************************************
                        001F  1834
                        001F  1835         SUBTEST                                          ;SUBTEST #1
                        0022
                        0022       ;////////////////////////////////////////////////////////////////////////////
                        0022  1836 ;+
                        0022  1837 ;SUBTEST #1
                        0022  1838 ;
                        0022  1839 ;       TEST XB0
                        0022  1840 ;-
                        0022  1841         INITIALIZE                                       ;INIT CPU
                        0024  1842         LOADDCS         5$,,15,1                         ;MODIFY DCS ADD 0F
                        002B  1843         LOOP            I,1,6                            ;CREATE A TEST LOOP
                        0032  1844         LDFIELD         1$,I,L,3$,31,62,LON              ;TEST PATTERN TO MICRO WORD
                        0043  1845         LOADDCS         3$,,06,1                         ;TEST PATTERN TO DCS
                        004A  1846         LDFIELD         2$,I,L,3$,31,62,LON              ;TEST PATTERN TO MICRO WORD
                        005B  1847         LOADDCS         3$,,10,1                         ;TEST PATTERN TO DCS
                        0062  1848         ERRLOOP                                          ;ERROR LOOP
                        0064  1849         FETCH           0                                ;START DCS PROGRAM
                        0069  1850         BRSTCLK         <^X0F>                           ;END DCS PROGRAM
                        006D  1851         CMPVBUS         6$                               ;TEST MSRC XB SIGNAL
                        0072  1852         IFERROR         ,,<MIC>                          ;SIGNAL NOT PRESENT
                        0078  1853         CMPREG          WHREG,1$,I,L,                    ;COMPARE RESULTS
                        0081  1854         IFERROR         ,<<MDR><ADD><PRK>>,<MIC>
                        008A  1855         ENDLOOP         I                                ;END TEST LOOP
                        008D  1856
                        008D  1857         SUBTEST                                          ;SUBTEST #2
                        0090
                        0090       ;////////////////////////////////////////////////////////////////////////////
                        0090  1858 ;+
```

```
                        0090  1859 ;SUBTEST #2
                        0090  1860 ;
                        0090  1861 ;       TEST XB1
                        0090  1862 ;-
                        0090  1863         INITIALIZE                                  ;INIT CPU
                        0092  1864         LOADDCS         4$,,15,1                    ;SOURCE XB1 TO RDM MICRO WORD
                        0099  1865         LOOP            I,1,6                       ;CREATE A TEST LOOP
                        00A0  1866         LDFIELD         1$,I,L,3$,31,62,LON         ;TEST PATTERN TO MICRO WORD
                        00B1  1867         LOADDCS         3$,,06,1                    ;TEST PATTERN TO DCS
                        00B8  1868         LDFIELD         2$,I,L,3$,31,62,LON         ;TEST PATTERN TO MICRO WORD
                        00C9  1869         LOADDCS         3$,,10,1                    ;TEST PATTERN TO DCS
                        00D0  1870         ERRLOOP                                     ;ERROR LOOP
                        00D2  1871         FETCH           0                           ;START DCS PROGRAM
                        00D7  1872         BRSTCLK         <^X10>                      ;END DCS PROGRAM
                        00DB  1873         CMPREG          WHREG,2$,I,L,               ;COMPARE RESULTS
                        00E4  1874         IFERROR         ,<<MDR><ADD><PRK>>,<MIC>
                        00ED  1875         ENDLOOP         I                           ;END TEST LOOP
                        00F0  1876         SKIP                                        ;GO TO NEXT TEST
                        00F7  1877
                        00F7  1878 BEGIN_TEST_DATA
                        00F7
                        00F7       ;-------------------------------------------------------------------------
                        00F7  1879
AAAAAAAA                00F7  1880 1$:     .LONG   ^XAAAAAAAA                          ;TEST PATTERNS FOR XB0
55555555                00FB  1881         .LONG   ^X55555555
33333333                00FF  1882         .LONG   ^X33333333
0F0F0F0F                0103  1883         .LONG   ^X0F0F0F0F
00FF00FF                0107  1884         .LONG   ^X00FF00FF
0000FFFF                010B  1885         .LONG   ^X0000FFFF
                        010F  1886
0000FFFF                010F  1887 2$:     .LONG   ^X0000FFFF                          ;TEST PATTERNS FOR XB1
00FF00FF                0113  1888         .LONG   ^X00FF00FF
0F0F0F0F                0117  1889         .LONG   ^X0F0F0F0F
33333333                011B  1890         .LONG   ^X33333333
55555555                011F  1891         .LONG   ^X55555555
AAAAAAAA                0123  1892         .LONG   ^XAAAAAAAA
                        0127  1893 3$:
U 06, 7700,7800,7FFF,FFFF,8470,1807  0127  1894 ;% 06   LONLIT_[0]                      ;TEST PATTERN TO LONGLIT REG
                        0132  1898
                        0132  1899 4$:
U 0F, 6701,4817,5924,0202,0470,1810  0132  1900 ;% 0F   RDM_WB,ISIZE[LONG],WB_M[XB.PC_PC+I]  ;SOURCE XB1 TO RDM
                        013D  1904
                        013D  1905 5$:
U 0F, 7300,C800,0364,0300,0470,1810  013D  1906 ;% 0F   NOP,STOP.CLOCK                  ;DCS ADD OF SUBTESTS #1
                     01 0148  1910 6$:     .BYTE   ^X1                                 ;TEST MSRC XB
                        0149  1911         VBUSDATA        <^X10>,1
                        014A  1912
                        014A  1913 BEGIN_CPU_DATA
                        0002  1914
                        0002  1915 DCS_DATA
                        0001  1916
U 00, 7700,C800,0364,0300,0470,1801  0001  1917 ;% 00   NOP
U 01, 7700,C800,5BE4,0300,6870,1802  000C  1921 ;% 01   MEMSCAR_R[TEMP0]                ;PHY/VIR REG ADD TO MEMSCAR
U 02, 7700,8800,5BE4,0304,6070,1803  0017  1925 ;% 02   MEMSCR_R[TEMP1]                 ;SET MME OFF
U 03, 7700,8800,5BE4,0308,6870,1804  0022  1929 ;% 03   MEMSCAR_R[TEMP2]                ;REF CACHE ADD TO MEMSCAR
```

```
U 04, 7700,C800,5BE4,030C,6070,1805  002D  1933 ;% 04   MEMSCR_R[TEMP3]                                  ;TURN CMI OFF
U 05, 7700,0800,5BE4,0310,4A70,1806  0038  1937 ;% 05   VA_R[TEMP4]                                     ;0'S TO VA REGISTER
U 06, 7700,7800,7FFF,FFFF,8470,1807  0043  1941 ;% 06   LONLIT_[0]                                     ;TEST PATTERN TO LONGLIT
U 07, 7701,8800,5BE4,02D4,5590,1808  004E  1945 ;% 07   WRITE.LONG R[LONLIT]                           ;TEST PATTERN TO CACHE ADD 0
U 08, 7700,4800,5BE4,03D4,5470,1809  0059  1949 ;% 08   WB_R[LONLIT],WCTRL/WDR_WB.UR                   ;HOLD DATA FOR EXTRA CYCLE
U 09, 7700,C800,0364,0300,4470,180A  0064  1953 ;% 09   VA_VA+4                                       ;INCREMENT VA BY 4
U 0A, 7700,7800,7FFF,FFFF,8470,180B  006F  1957 ;% 0A   LONLIT_[0]                                     ;TEST PATTERN TO LONGLIT
U 0B, 7701,0800,5BE4,02D4,5590,180C  007A  1961 ;% 0B   WRITE.LONG R[LONLIT]                           ;TEST PATTERN TO CACHE ADD 4
U 0C, 7700,C800,5BE4,03D4,5470,180D  0085  1965 ;% 0C   WB_R[LONLIT],WCTRL/WDR_WB.UR                   ;HOLD DATA FOR EXTRA CYCLE
U 0D, 7700,0800,5BE4,0310,4870,180E  0090  1969 ;% 0D   PC_R[TEMP4]                                    ;0'S TO PC
U 0E, 6601,C817,5924,0202,0470,180F  009B  1973 ;% 0E   RDM_WB,ISIZE[LONG],WB_M[XB.PC_PC+I],STROBE.V.BUS ;READ XB0
U 0F, 7300,C800,0364,0300,0470,1810  00A6  1977 ;% 0F   NOP,STOP.CLOCK                                 ;STOP CPU CLOCK
U 10, 7300,4800,0364,0300,0470,1811  00B1  1981 ;% 10   NOP,STOP.CLOCK                                 ;STOP CPU CLOCK
                                     00BC  1985
                                     00BC  1986 RTEMP_DATA
                                     0001  1987
                           00000000  0001  1988          .LONG   ^X00000000                          ;PHY/VIR REG ADDRESS
                           00000000  0005  1989          .LONG   ^X00000000                          ;SET MME OFF
                           0E000000  0009  1990          .LONG   ^X0E000000                          ;REF CACHE ONLY ADDRESS
                           01000000  000D  1991          .LONG   ^X01000000                          ;TURN CMI OFF
                           00000000  0011  1992          .LONG   ^X00000000                          ;ADDRESS FOR VA & PC
                                     0015  1993
                                     0015  1994 END_CPU_DATA
                                     014A  1995
                                     014A  1996 END_TEST_DATA
                                     014A
                                     014A          ;-----------------------------------------------------------------------
                                     014A  1997
                                     014A  1998 ENDTEST
```

```
                        001E        .SBTTL  "TEST   06B     Prefetch Incrementer Test
                        001E  2000 ;*******************************************************************************
                        001E  2001 ;++
                        001E  2002 ;
                        001E  2003 ; FUNCTIONAL DESCRIPTION
                        001E  2004 ;
                        001E  2005 ;       This test will examine the combinatorial logic that implements the
                        001E  2006 ;       PC+4 increment used in instruction prefetch. The instruction prefetch
                        001E  2007 ;       operation must be used to implement this test because the PC+4
                        001E  2008 ;       increment logic cannot be sourced directly onto the MBUS. Therefore
                        001E  2009 ;       it is necessary to initiate prefetch by loading the PC with an address
                        001E  2010 ;       and then comparing the data written into XB0 and XB1 with the expected
                        001E  2011 ;       data stored in the RDM memory. If the two are different an error has
                        001E  2012 ;       occurred.
                        001E  2013 ;
                        001E  2014 ;       NOTE:   IF THIS TEST FAILS AND THE MIC MODULE IS NOT KNOWN GOOD
                        001E  2015 ;               THEN YOU SHOULD RUN THE MIC TESTS AT THIS TIME TO VERIFY
                        001E  2016 ;               IT IS OPERATIONAL.
                        001E  2017 ;
                        001E  2018 ; TEST ALGORITHM
                        001E  2019 ;
                        001E  2020 ;       1.  Initialize the CPU to clear the CACHE and TB.
                        001E  2021 ;       2.  Disable the CMI.
                        001E  2022 ;       3.  Enable memory management, TB and CACHE.
                        001E  2023 ;       4.  Load the cache with the valid PC data code at addresses specified
                        001E  2024 ;           by the VA.
                        001E  2025 ;       5.  Load the cache with the valid PC+4 data code at addresses
                        001E  2026 ;           specified by the +4 increments of the VA.
                        001E  2027 ;       6.  Initiate instruction prefetch by loading the PC with an address.
                        001E  2028 ;       7.  Wait for prefetch to finish.
                        001E  2029 ;       8.  Read XB0 to the RDM and check for the PC valid data code.
                        001E  2030 ;           If not present, signal an error.
                        001E  2031 ;       9.  Increment the PC by four.
                        001E  2032 ;       10. Read XB1 to the RDM and check for the PC+4 valid data code.
                        001E  2033 ;           If not present, signal an error.
                        001E  2034 ;       11. Continue steps 6-10 for all 32 address test patterns.
                        001E  2035 ;       12. Test completed.
                        001E  2036 ;
                        001E  2037 ; DATA PATTERNS
                        001E  2038 ;
                        001E  2039 ;       Thirty-one address patterns are required to fully test the increment
                        001E  2040 ;       logic. They are listed below:
                        001E  2041 ;
                        001E  2042 ;               LOOP COUNT          PC  ADDRESS             PC+4  ADDRESS
                        001E  2043 ;
                        001E  2044 ;       1           1           F F F F F F F 8         F F F F F F F C
                        001E  2045 ;       2           2           F F F F F F F 4         F F F F F F F 8
                        001E  2046 ;       3           3           F F F F F F E C         F F F F F F F 0
                        001E  2047 ;       4           4           F F F F F F D C         F F F F F F E 0
                        001E  2048 ;       5           5           F F F F F F B C         F F F F F F C 0
                        001E  2049 ;       6           6           F F F F F F 7 C         F F F F F F 8 0
                        001E  2050 ;       7           7           F F F F F E F C         F F F F F F 0 0
                        001E  2051 ;       8           8           F F F F F D F C         F F F F F E 0 0
                        001E  2052 ;       9           9           F F F F F B F C         F F F F F C 0 0
                        001E  2053 ;       10          A           F F F F F 7 F C         F F F F F 8 0 0
```

```
001E  2054 ;       11          B          F F F F E F F C          F F F F F 0 0 0
001E  2055 ;       12          C          F F F F D F F C          F F F F E 0 0 0
001E  2056 ;       13          D          F F F F B F F C          F F F F C 0 0 0
001E  2057 ;       14          E          F F F F 7 F F C          F F F F 8 0 0 0
001E  2058 ;       15          F          F F F E F F F C          F F F F 0 0 0 0
001E  2059 ;       16          10         F F F D F F F C          F F F E 0 0 0 0
001E  2060 ;       17          11         F F F B F F F C          F F F C 0 0 0 0
001E  2061 ;       18          12         F F F 7 F F F C          F F F 8 0 0 0 0
001E  2062 ;       19          13         F F E F F F F C          F F F 0 0 0 0 0
001E  2063 ;       20          14         F F D F F F F C          F F E 0 0 0 0 0
001E  2064 ;       21          15         F F B F F F F C          F F C 0 0 0 0 0
001E  2065 ;       22          16         F F 7 F F F F C          F F 8 0 0 0 0 0
001E  2066 ;       23          17         F E F F F F F C          F F 0 0 0 0 0 0
001E  2067 ;       24          18         F D F F F F F C          F E 0 0 0 0 0 0
001E  2068 ;       25          19         F B F F F F F C          F C 0 0 0 0 0 0
001E  2069 ;       26          1A         F 7 F F F F F C          F 8 0 0 0 0 0 0
001E  2070 ;       27          1B         E F F F F F F C          F 0 0 0 0 0 0 0
001E  2071 ;       28          1C         D F F F F F F C          E 0 0 0 0 0 0 0
001E  2072 ;       29          1D         B F F F F F F C          C 0 0 0 0 0 0 0
001E  2073 ;       30          1E         7 F F F F F F C          8 0 0 0 0 0 0 0
001E  2074 ;       31          1F         F F F F F F F C          0 0 0 0 0 0 0 0
001E  2075 ;
001E  2076 ;       Cache to XB valid data codes:
001E  2077 ;
001E  2078 ;               PC valid data code      XB0 = A A A A A A A A
001E  2079 ;               PC+4 valid data code    XB1 = 5 5 5 5 5 5 5 5
001E  2080 ;
001E  2081 ; LOGIC DESCRIPTION
001E  2082 ;
001E  2083 ;       This test consists of two parts. The first part examines the low order
001E  2084 ;       two bits of address to make sure they are not stuck at one or zero.
001E  2085 ;       The second part will test the carry logic within the incrementer by
001E  2086 ;       floating a zero across all bits in the PC while rippling a carry into
001E  2087 ;       the zero from the incrementer.
001E  2088 ;
001E  2089 ; ASSUMPTIONS
001E  2090 ;
001E  2091 ;       This test assumes that the DPM, WBUS and MBUS are functional. Since
001E  2092 ;       this test requires the use of the instuction prefetch mechanism, all
001E  2093 ;       tests on the VA, PC, the two execution buffers, cache, translation
001E  2094 ;       buffer and WDR must also have been successfully completed.
001E  2095 ;
001E  2096 ; ERROR DESCRIPTION
001E  2097 ;
001E  2098 ;       Below is a table of possible error symptoms and their most likely
001E  2099 ;       causes:
001E  2100 ;
001E  2101 ;               Symptom                         Probable Cause
001E  2102 ;       +-----------------------+-----------------------------------------+
001E  2103 ;       | An error is called out| A defective PRK gate array is causing   |
001E  2104 ;       | regardless of whether | the incrementer or PC from being sent   |
001E  2105 ;       | the expected data was | out on the address lines.               |
001E  2106 ;       | AAAAAAAA or 55555555. |                                         |
001E  2107 ;       +-----------------------+-----------------------------------------+
001E  2108 ;       | An error is called out| A problem with either the logic used    |
```

```
001E  ;        | when the expected data | to load the PC or the PC register     |
001E  ;        | was AAAAAAAA.          | itself. The PC register diagnostic    |
001E  ;        |                        | should be run to isolate the problem. |
001E  ;        +------------------------+---------------------------------------+
001E  ;        | An error is called out |                                       |
001E  ;        | with the expected data | The ADD gate array used to generate   |
001E  ;        | being 55555555 and the | bits <07:00> of the memory address.   |
001E  ;        | loop count was in the  |                                       |
001E  ;        | range of 1 to 6 hex.   |                                       |
001E  ;        +------------------------+---------------------------------------+
001E  ;        | An error is called out |                                       |
001E  ;        | with the expected data | The ADD gate array used to generate   |
001E  ;        | being 55555555 and the | bits <15:08> of the memory address.   |
001E  ;        | loop count was in the  |                                       |
001E  ;        | range of 7 to E hex.   |                                       |
001E  ;        +------------------------+---------------------------------------+
001E  ;        | An error is called out |                                       |
001E  ;        | with the expected data | The ADD gate array used to generate   |
001E  ;        | being 55555555 and the | bits <23:16> of the memory address.   |
001E  ;        | loop count was in the  |                                       |
001E  ;        | range of 0F to 16 hex. |                                       |
001E  ;        +------------------------+---------------------------------------+
001E  ;        | An error is called out |                                       |
001E  ;        | with the expected data | The ADD gate array used o generate    |
001E  ;        | being 55555555 and the | bits <31:24> of the memory address.   |
001E  ;        | loop count was in the  |                                       |
001E  ;        | range of 17 to 1E hex. |                                       |
001E  ;        +------------------------+---------------------------------------+
001E  ;        | An error is called out | A problem with the carry out from the |
001E  ;        | when the loop count    | incrementer not being passed onto the |
001E  ;        | was equal to 1F hex.   | next higher order gate array. The     |
001E  ;        |                        | carry out - carry in connection       |
001E  ;        |                        | between the gate arrays should be     |
001E  ;        |                        | checked.                              |
001E  ;        +------------------------+---------------------------------------+
001E  ;
001E  ;        On error this test will call out the ADD and PRK gate array on the
001E  ;        MIC module. The following information will also by typed out:
001E  ;
001E  ;        Expected XB data:
001E  ;        Received XB data:
001E  ;        Loop Count:
001E  ;
001E  ;--
001E  ;*******************************************************************************
001E  ;///////////////////////////////////////////////////////////////////////////////
001E  ;+
001E  ; Memory control register assignments
001E  ;
001E  ;        Disables the CMI and enables memory management, cache and TB.
001E  ;-
001E          INITIALIZE
0020          LOADDCS         6$,,0,18
0027          FETCH           0
002C          BRSTCLK         <^X11>
```

```
                        0030  2164 ;
                        0030  2165 ; This code invalidates the Translation Buffers.
                        0030  2166 ;
                        0030  2167          LOADDCS         7$,,0,4
                        0037  2168          FETCH           0
                        003C  2169          BRSTCLK         3
                        0040  2170          LOADDCS         8$,,0,4
                        0047  2171          LOOP            I,1,128
                        004E  2172          FETCH           0
                        0053  2173          BRSTCLK         3
                        0057  2174          ENDLOOP         I
                        005A  2175 ;
                        005A  2176 ; This code is used to invalidate all of the Cache.
                        005A  2177 ;
                        005A  2178          LOADDCS         7$,,0,4
                        0061  2179          FETCH           0
                        0066  2180          BRSTCLK         3
                        006A  2181          LOADDCS         9$,,0,4
                        0071  2182          LOOP            I,1,1024
                        0078  2183          FETCH           0
                        007D  2184          BRSTCLK         3
                        0081  2185          ENDLOOP         I
                        0084  2186 ;
                        0084  2187 ; This code examines the prefetch incrementer.
                        0084  2188 ;
                        0084  2189          LOADDCS         10$,,0,<^X1F>
                        008B  2190          LOOP            I,1,31
                        0092  2191          LOADREG         WDREG,1$,I,LONG
                        009A  2192          EXPUTRAP
                        009C  2193          ERRLOOP
                        009E  2194          FETCH           0
                        00A3  2195          BRSTCLK         <^X16>
                        00A7  2196          CMPREGMSK       CSTRP,4$,,5$,,WORD
                        00B3  2197          IFERROR         ,<<ADD>,<PRK>>,<MIC>
                        00BB  2198          CMPREG          WHREG,2$,,LONG
                        00C4  2199          IFERROR         ,<<ADD>,<PRK>>,<MIC>
                        00CC  2200          FETCH           <^X14>
                        00D1  2201          BRSTCLK         <^X16>
                        00D5  2202          CMPREGMSK       CSTRP,4$,,5$,,WORD
                        00E1  2203          IFERROR         ,<<ADD>,<PRK>>,<MIC>
                        00E9  2204          CMPREG          WHREG,3$,,LONG
                        00F2  2205          IFERROR         ,<<ADD>,<PRK>>,<MIC>
                        00FA  2206          FETCH           <^X17>
                        00FF  2207          BRSTCLK         <^X1E>
                        0103  2208          ENDLOOP         I
                        0106  2209          SKIP
                        010D  2210
                        010D  2211 BEGIN_TEST_DATA
                        010D
                        010D          ;-----------------------------------------------------------------------
                        010D  2212
FFFFFFF8                010D  2213 1$:      .LONG    ^X FFFFFFF8                ; PC addresses
FFFFFFF4                0111  2214          .LONG    ^X FFFFFFF4
FFFFFFEC                0115  2215          .LONG    ^X FFFFFFEC
FFFFFFDC                0119  2216          .LONG    ^X FFFFFFDC
```

```
                FFFFFFBC  011D  2217          .LONG   ^X FFFFFFBC
                FFFFFF7C  0121  2218          .LONG   ^X FFFFFF7C
                FFFFFEFC  0125  2219          .LONG   ^X FFFFFEFC
                FFFFDFFC  0129  2220          .LONG   ^X FFFFDFFC
                FFFFBFFC  012D  2221          .LONG   ^X FFFFBFFC
                FFFF7FFC  0131  2222          .LONG   ^X FFFF7FFC
                FFFEFFFC  0135  2223          .LONG   ^X FFFEFFFC
                FFFDFFFC  0139  2224          .LONG   ^X FFFDFFFC
                FFFBFFFC  013D  2225          .LONG   ^X FFFBFFFC
                FFF7FFFC  0141  2226          .LONG   ^X FFF7FFFC
                FFEFFFFC  0145  2227          .LONG   ^X FFEFFFFC
                FFDFFFFC  0149  2228          .LONG   ^X FFDFFFFC
                FFBFFFFC  014D  2229          .LONG   ^X FFBFFFFC
                FF7FFFFC  0151  2230          .LONG   ^X FF7FFFFC
                FEFFFFFC  0155  2231          .LONG   ^X FEFFFFFC
                FDFFFFFC  0159  2232          .LONG   ^X FDFFFFFC
                FBFFFFFC  015D  2233          .LONG   ^X FBFFFFFC
                F7FFFFFC  0161  2234          .LONG   ^X F7FFFFFC
                EFFFFFFC  0165  2235          .LONG   ^X EFFFFFFC
                DFFFFFFC  0169  2236          .LONG   ^X DFFFFFFC
                BFFFFFFC  016D  2237          .LONG   ^X BFFFFFFC
                7FFFFFFC  0171  2238          .LONG   ^X 7FFFFFFC
                FFFFFFFC  0175  2239          .LONG   ^X FFFFFFFC
                          0179  2240
                AAAAAAAA  0179  2241 2$:      .LONG   ^X AAAAAAAA            ; PC address valid data code
                55555555  017D  2242 3$:      .LONG   ^X 55555555            ; PC+4 address valid data code
                          0181  2243
                    1816  0181  2244 4$:      .WORD   ^X 1816                ; EXPECTED CS ADDRESS
                    3FFF  0183  2245 5$:      .WORD   ^X 3FFF                ; CS ADDRESS MASK
                          0185  2246
                          0185  2247 6$:
U 00, 7700,C800,0364,0300,0470,1801   0185  2248 ;* 00    NOP
U 01, 7700,3800,78FF,FFFF,8470,1802   0190  2249 ;* 01    LONLIT_[0E000000]      ; Address of CMI enable/disable MEMSCR
U 02, 7700,4800,5BE4,03D4,6870,1803   0199  2250 ;* 02    MEMSCAR_R[LONLIT]
U 03, 7700,3800,7F7F,FFFF,8470,1804   01A4  2251 ;* 03    LONLIT_[01000000]      ; Disables CMI
U 04, 7700,C800,5BE4,03D4,6070,1805   01AF  2252 ;* 04    MEMSCR_R[LONLIT]
U 05, 7700,F800,7FFF,FFFF,8470,1806   01BA  2253 ;* 05    LONLIT_[0]             ; Address of MM enable/disable MEMSCR
U 06, 7700,C800,5BE4,03D4,6870,1807   01C5  2254 ;* 06    MEMSCAR_R[LONLIT]
U 07, 7700,3800,7F7F,FFFF,8470,1808   01D0  2255 ;* 07    LONLIT_[01000000]      ; Enables memory management
U 08, 7700,C800,5BE4,03D4,6070,1809   01DB  2256 ;* 08    MEMSCR_R[LONLIT]
U 09, 7700,F800,7E7F,FFFF,8470,180A   01E6  2257 ;* 09    LONLIT_[03000000]      ; Address of TB enable/disable MEMSCR
U 0A, 7700,C800,5BE4,03D4,6870,180B   01F1  2258 ;* 0A    MEMSCAR_R[LONLIT]
U 0B, 7700,F800,7AFF,FFFF,8470,180C   01FC  2259 ;* 0B    LONLIT_[0A000000]      ; Enables TB group 0 only
U 0C, 7700,4800,5BE4,03D4,6070,180D   0207  2260 ;* 0C    MEMSCR_R[LONLIT]
U 0D, 7700,7800,7CFF,FFFF,8470,180E   0212  2261 ;* 0D    LONLIT_[06000000]      ; Address of CACHE enable/disable MEMSCR
U 0E, 7700,4800,5BE4,03D4,6870,180F   021D  2262 ;* 0E    MEMSCAR_R[LONLIT]
U 0F, 7700,7800,7FFF,FFFF,8470,1810   0228  2263 ;* 0F    LONLIT_[0]             ; Enables CACHE
U 10, 7700,C800,5BE4,03D4,6070,1811   0233  2264 ;* 10    MEMSCR_R[LONLIT]
U 11, 7300,4800,0364,0300,0470,1812   023E  2265 ;* 11    STOP.CLOCK
                          0249  2266
                          0249  2267 7$:
```

```
U 00. 7700.C800.0364.0300.0470.1801  025B  2326 ;% 00   NOP
U 01. 7700.3800.7FFF.FDFF.8470.1802  0266  2330 ;% 01   LONLIT_[400]
U 02. 7700.0800.5BE6.03D8.4A70.1803  0271  2334 ;% 02   D_VA_0
U 03. 7300.C800.0364.0300.0470.1804  027C  2338 ;% 03   STOP.CLOCK
                                     0287  2342
                                     0287  2343 8$:
U 00. 7700.C800.0364.0300.0470.1801  0287  2344 ;% 00   NOP
U 01. 7700.C800.0224.03D8.5360.1802  0292  2348 ;% 01   CLRTB.VA_D                ; WCTRL/CLRTB.VA_WB,WB_D
U 02. 7700.4800.0212.03D4.0470.1803  029D  2352 ;% 02   D_D+R[LONLIT]
U 03. 7300.C800.0364.0300.0470.1804  02A8  2356 ;% 03   STOP.CLOCK
                                     02B3  2360
                                     02B3  2361 9$:
U 00. 7700.C800.0364.0300.0470.1801  02B3  2362 ;% 00   NOP
U 01. 7700.081B.0024.03D8.5A70.1802  02BE  2366 ;% 01   CLRCH.VA_VA               ; WCTRL/CLRCH.VA_WB,WB_VA
U 02. 7700.C800.0364.0300.4470.1803  02C9  2370 ;% 02   VA_VA+4
U 03. 7300.C800.0364.0300.0470.1804  02D4  2374 ;% 03   STOP.CLOCK
                                     02DF  2378
                                     02DF  2379 10$:
U 00. 7700.C800.0364.0300.0470.1801  02DF  2380 ;% 00   NOP
U 01. 5700.C800.0364.0300.4A70.1802  02EA  2384 ;% 01   VA_RDM
U 02. 7700.B800.7FFF.FF5B.8470.1803  02F5  2388 ;% 02   LONLIT_[00000148]
U 03. 7700.481B.5BE4.03D4.5070.1804  0300  2392 ;% 03   TB_R[LONLIT]
U 04. 7700.F800.2AAA.AAAA.8470.1805  030B  2396 ;% 04   LONLIT_[0AAAAAAAA]
U 05. 7701.0800.5BE4.02D4.5590.1806  0316  2400 ;% 05   WRITE.LONG R[LONLIT]
U 06. 7700.C800.5BE4.03D4.5470.1807  0321  2404 ;% 06   WDR_UNROT(R[LONLIT])
U 07. 7700.4800.0364.0300.4470.1808  032C  2408 ;% 07   VA_VA+4
U 08. 7700.B800.7FFF.FF5B.8470.1809  0337  2412 ;% 08   LONLIT_[00000148]
U 09. 7700.C81B.5BE4.03D4.5070.180A  0342  2416 ;% 09   TB_R[LONLIT]
U 0A. 7700.B800.5555.5555.0470.180B  034D  2420 ;% 0A   LONLIT_[55555555]
U 0B. 7701.0800.5BE4.02D4.5590.180C  0358  2424 ;% 0B   WRITE.LONG R[LONLIT]
U 0C. 7700.C800.5BE4.03D4.5470.180D  0363  2428 ;% 0C   WDR_UNROT(R[LONLIT])
U 0D. 7700.4800.0364.0300.4470.180E  036E  2432 ;% 0D   VA_VA+4
U 0E. 7700.B800.7FFF.FF5B.8470.180F  0379  2436 ;% 0E   LONLIT_[00000148]
U 0F. 7700.481B.5BE4.03D4.5070.1810  0384  2440 ;% 0F   TB_R[LONLIT]
U 10. 7700.3800.0000.0000.0470.1811  038F  2444 ;% 10   LONLIT_[0FFFFFFFF]
U 11. 7701.0800.5BE4.02D4.5590.1812  039A  2448 ;% 11   WRITE.LONG R[LONLIT]
U 12. 7700.C800.5BE4.03D4.5470.1813  03A5  2452 ;% 12   WDR_UNROT(R[LONLIT])
U 13. 5700.C800.0364.0300.4870.1814  03B0  2456 ;% 13   PC_RDM
U 14. 7700.C800.0364.0300.0470.1815  03BB  2460 ;% 14   NOP
U 15. 6701.4817.0024.02DA.0470.1816  03C6  2464 ;% 15   PC_PC+4 RDM_XB
U 16. 7300.4800.0364.0300.0470.1817  03D1  2468 ;% 16   STOP.CLOCK
U 17. 7700.4800.0364.0300.0470.1818  03DC  2472 ;% 17   NOP
U 18. 5700.C800.0364.0300.4A70.1819  03E7  2476 ;% 18   VA_RDM
U 19. 7701.8800.5BE4.02D8.5590.181A  03F2  2480 ;% 19   WRITE.LONG R[ZERO]
U 1A. 7700.4800.5BE4.03D8.5470.181B  03FD  2484 ;% 1A   WDR_UNROT(R[ZERO])
U 1B. 7700.4800.0364.0300.4470.181C  0408  2488 ;% 1B   VA_VA+4
U 1C. 7701.0800.5BE4.02D8.5590.181D  0413  2492 ;% 1C   WRITE.LONG R[ZERO]
U 1D. 7700.4800.5BE4.03D8.5470.181E  041E  2496 ;% 1D   WDR_UNROT(R[ZERO])
U 1E. 7300.C800.0364.0300.0470.181F  0429  2500 ;% 1E   STOP.CLOCK
                                     0434  2504
                                     0434  2505 END_TEST_DATA
                                     0434
                                     0434       ;-----------------------------------------------------------------------------
                                     0434  2506
                                     0434  2507 ENDTEST
```

```
ZZ-ECKAB-9.2    "TEST 06B Prefetch Incrementer Test                13-MAY-1987       Fiche 3 Frame F9          Sequence 521
ECKAB                          VAX-11/750 DPM MICRO DIAGNOSTICS       11-DEC-1985 09:34:06  VAX/VMS Macro V04-00        Page  6
V09.01                         "TEST 06B Prefetch Incrementer Test    11-DEC-1985 09:33:26  [VAX750.DPM]ECKAB3.MAR;1          (1

%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
                        0027        .SBTTL  "TEST   06C     Test Execution Buffer Rotate Logic
0027  2509 ;*****************************************************************************
0027  2510 ;++
0027  2511 ;
0027  2512 ; FUNCTIONAL DESCRIPTION:
0027  2513 ;
0027  2514 ;       THIS TEST CONSISTS OF 2 SUBTESTS TO CHECK THE OPERATION OF THE ROTATE
0027  2515 ;       LOGIC ON THE OUTPUT OF XB0 AND XB1.
0027  2516 ;       NOTE:
0027  2517 ;               IF THIS TEST FAILS AND THE MIC MODULE IS NOT KNOWN GOOD;
0027  2518 ;               THEN YOU SHOULD RUN THE MIC TESTS AT THIS TIME TO VERIFY
0027  2519 ;               IT IS OPERATIONAL.
0027  2520 ;
0027  2521 ;
0027  2522 ;
0027  2523 ; TEST ALGORITHM:
0027  2524 ;
0027  2525 ;       1.  CREATE A TEST LOOP
0027  2526 ;       2.  LOAD DCS WITH MICRO WORDS FOR THIS TEST
0027  2527 ;       3.  EXECUTE DCS PROGRAM
0027  2528 ;               1.  LOAD ADDRESS OF PHYSICAL/VIRTUAL ADDRESSING REGISTER INTO
0027  2529 ;                   MEMORY STATUS AND CONTROL ADDRESS REGISTER (MEMSCAR)
0027  2530 ;               2.  SET MEMORY MANAGEMENT OFF
0027  2531 ;               3.  LOAD ADDRESS OF REFERENCE CACHE ONLY REGISTER INTO MEMSCAR
0027  2532 ;               4.  SET CMI OFF
0027  2533 ;               5.  LOAD VIRTUAL ADDRESS REGISTER WITH 0'S
0027  2534 ;               6.  PUT TEST PATTERN #1 INTO LONGLIT REGISTER
0027  2535 ;               7.  WRITE TEST PATTERN IN LONGLIT TO CACHE ADDRESS 0
0027  2536 ;               8.  INCREMENT VIRTUAL ADDRESS REGISTER BY 4
0027  2537 ;               9.  PUT TEST PATTERN #2 INTO LONGLIT REGISTER
0027  2538 ;              10.  WRITE TEST PATTERN IN LONGLIT TO CACHE ADDRESS 4
0027  2539 ;              11.  INCREMENT VIRTUAL ADDRESS REGISTER BY 4
0027  2540 ;              12.  PUT TEST PATTERN #3 INTO LONGLIT REGISTER
0027  2541 ;              13.  WRITE TEST PATTERN IN LONGLIT TO CACHE ADDRESS 8
0027  2542 ;              14.  LOAD PROGRAM COUNTER WITH A PREFETCH ADDRESS
0027  2543 ;              12.  READ DESIRED TEST PATTERN FROM XB'S TO RDM
0027  2544 ;       4.  COMPARE RESULTS (EXPECTED AND RECEIVED)
0027  2545 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST ITERATIONS
0027  2546 ;
0027  2547 ;
0027  2548 ; TEST PATTERNS:
0027  2549 ;
0027  2550 ;       ITERATION       PATTERN 3       PATTERN 2       PATTERN 1
0027  2551 ;
0027  2552 ;           1           00000000        FFFFFF00        000000FF
0027  2553 ;           2           00000000        000000FF        FFFFFF00
0027  2554 ;           3           000000FF        FFFF0000        0000FF00
0027  2555 ;           4           00000000        0000FFFF        FFFF0000
0027  2556 ;           5           0000FFFF        FF000000        00FF0000
0027  2557 ;           6           00000000        00FFFFFF        FF000000
0027  2558 ;           7           00FFFFFF        00000000        FF000000
0027  2559 ;           8           00000000        FFFFFFFF        00000000
0027  2560 ;
0027  2561 ;
0027  2562 ; LOGIC DESCRIPTION:
```

```
                        0027  2563 ;
                        0027  2564 ;       THIS TEST IS DIVIDED INTO 2 SUBTESTS, EACH OF WHICH IS ALMOST
                        0027  2565 ;       IDENTICAL.  SUBTEST 1 LOOKS AT THE OUTPUT OF XB0 AND SUBTEST 2
                        0027  2566 ;       LOOKS AT THE OUTPUT OF XB1.  TEST PATTERNS AND THEIR COMPLIMENTS
                        0027  2567 ;       ARE SHIFTED FROM ALL BYTE POSITIONS OF XB0 AND XB1 VIA THE EXECUTION
                        0027  2568 ;       BUFFER ROTATOR TO THE WBUS.  THE "MDR" GATE ARRAY IS YOUR BEST FIRST
                        0027  2569 ;       CHOICE TO SWAP IN THE EVENT OF ERRORS.  SECOND AND THIRD CHOICE SHOULD
                        0027  2570 ;       BE "ADD", WHICH CONTROLS THE ADDRESSING, AND "PRK", WHICH CONTROLS THE
                        0027  2571 ;       PREFETCH MECHANISM.
                        0027  2572 ;
                        0027  2573 ;
                        0027  2574 ; ASSUMPTIONS:
                        0027  2575 ;
                        0027  2576 ;       THIS TEST ASSUMES THE WBUS, CACHE, VA, PC, AND MBUS ARE ALL
                        0027  2577 ;       OPERATIONAL.
                        0027  2578 ;
                        0027  2579 ;
                        0027  2580 ; ERROR DESCRIPTION:
                        0027  2581 ;
                        0027  2582 ;       EXPECTED XB DATA
                        0027  2583 ;       RECEIVED XB DATA
                        0027  2584 ;       LOOP COUNT
                        0027  2585 ;
                        0027  2586 ;--
                        0027  2587 ;*******************************************************************************
                        0027  2588 ;///////////////////////////////////////////////////////////////////////////////
                        0027  2589 ;+
                        0027  2590 ;SUBTEST #1
                        0027  2591 ;
                        0027  2592 ;       TEST EXECUTION BUFFER ROTATE LOGIC USING XB0
                        0027  2593 ;-
                        0027  2594          SUBTEST                                         ;SUBTEST #1
                        002A
                        002A        ;///////////////////////////////////////////////////////////////////////////////
                        002A  2595          INITIALIZE                                      ;INIT CPU
                        002C  2596          LOADDCS         10$,,19,1                       ;MODIFY DCS ADD 19
                        0033  2597          LOOP            I,1,8                           ;CREATE A TEST LOOP
                        003A  2598          LDFIELD         2$,I,L,1$,31,62,LON             ;TEST PATTERN TO MICRO WORD
                        004B  2599          LOADDCS         1$,,06,1                        ;TEST PATTERN #1 TO DCS
                        0052  2600          LDFIELD         3$,I,L,1$,31,62,LON             ;TEST PATTERN TO MICRO WORD
                        0063  2601          LOADDCS         1$,,10,1                        ;TEST PATTERN #2 TO DCS
                        006A  2602          LDFIELD         4$,I,L,1$,31,62,LON             ;TEST PATTERN TO MICRO WORD
                        007B  2603          LOADDCS         1$,,14,1                        ;TEST PATTERN #3 TO DCS
                        0082  2604          LDFIELD         5$,I,B,6$,34,39                 ;RTEMP FOR PC
                        0093  2605          LOADDCS         6$,,17,1                        ;STARTING PC TO DCS
                        009A  2606          ERRLOOP                                         ;ERROR LOOP
                        009C  2607          FETCH           0                               ;START DCS PROGRAM
                        00A1  2608          BRSTCLK         <^X13>                          ;END DCS PROGRAM
                        00A5  2609          CMPREG          WHREG,7$,I,L,                   ;COMPARE RESULTS
                        00AE  2610          IFERROR         ,<<MDR><ADD><PRK>>,<MIC>
                        00B7  2611          ENDLOOP         I                               ;END TEST LOOP
                        00BA  2612
                        00BA  2613
                        00BA  2614
                        00BA  2615 ;///////////////////////////////////////////////////////////////////////////////
```

```
                        00BA  2616 ;*
                        00BA  2617 ;SUBTEST #2
                        00BA  2618 ;
                        00BA  2619 ;       TEST EXECUTION BUFFER ROTATE LOGIC USING XB1
                        00BA  2620 ;-
                        00BA  2621         SUBTEST                                      ;SUBTEST #2
                        00BD
                        00BD       ;////////////////////////////////////////////////////////////////////////
                        00BD  2622         INITIALIZE                                   ;INIT CPU
                        00BF  2623         LOADDCS         8$,,19,1                     ;SOURCE XB1 TO RDM MICRO WORD
                        00C6  2624         LOOP            I,1,8                        ;CREATE A TEST LOOP
                        00CD  2625         LDFIELD         2$,I,L,1$,31,62,LON          ;TEST PATTERN TO MICRO WORD
                        00DE  2626         LOADDCS         1$,,06,1                     ;TEST PATTERN #1 TO DCS
                        00E5  2627         LDFIELD         3$,I,L,1$,31,62,LON          ;TEST PATTERN TO MICRO WORD
                        00F6  2628         LOADDCS         1$,,10,1                     ;TEST PATTERN #2 TO DCS
                        00FD  2629         LDFIELD         4$,I,L,1$,31,62,LON          ;TEST PATTERN TO MICRO WORD
                        010E  2630         LOADDCS         1$,,14,1                     ;TEST PATTERN #3 TO DCS
                        0115  2631         LDFIELD         5$,I,B,6$,34,39              ;RTEMP FOR PC
                        0126  2632         LOADDCS         6$,,17,1                     ;STARTING PC TO DCS
                        012D  2633         ERRLOOP                                      ;ERROR LOOP
                        012F  2634         FETCH           0                            ;START DCS PROGRAM
                        0134  2635         BRSTCLK         <^X14>                       ;END DCS PROGRAM
                        0138  2636         CMPREG          WHREG,9$,I,L,                ;COMPARE RESULTS
                        0141  2637         IFERROR         ,<<MDR><ADD><PRK>>,<MIC>
                        014A  2638         ENDLOOP         I                            ;END TEST LOOP
                        014D  2639         SKIP
                        0154  2640
                        0154  2641 BEGIN_TEST_DATA
                        0154
                        0154       ;-------------------------------------------------------------------------
                        0154  2642
                        0154  2643 1$:
U 06, 7700,7800,7FFF,FFFF,8470,1807  0154  2644 ;* 06   LONLIT_[0]                      ;TEST PATTERNS TO LONGLIT REG
                        015F  2648
                000000FF  015F  2649 2$:     .LONG   ^X000000FF                       ;TEST PATTERN #1
                FFFFFF00  0163  2650         .LONG   ^XFFFFFF00
                0000FF00  0167  2651         .LONG   ^X0000FF00
                FFFF0000  016B  2652         .LONG   ^XFFFF0000
                00FF0000  016F  2653         .LONG   ^X00FF0000
                FF000000  0173  2654         .LONG   ^XFF000000
                FF000000  0177  2655         .LONG   ^XFF000000
                00000000  017B  2656         .LONG   ^X00000000
                        017F  2657
                FFFFFF00  017F  2658 3$:     .LONG   ^XFFFFFF00                       ;TEST PATTERN #2
                000000FF  0183  2659         .LONG   ^X000000FF
                FFFF0000  0187  2660         .LONG   ^XFFFF0000
                0000FFFF  018B  2661         .LONG   ^X0000FFFF
                FF000000  018F  2662         .LONG   ^XFF000000
                00FFFFFF  0193  2663         .LONG   ^X00FFFFFF
                00000000  0197  2664         .LONG   ^X00000000
                FFFFFFFF  019B  2665         .LONG   ^XFFFFFFFF
                        019F  2666
                00000000  019F  2667 4$:     .LONG   ^X00000000                       ;TEST PATTERN #3
                00000000  01A3  2668         .LONG   ^X00000000
                000000FF  01A7  2669         .LONG   ^X000000FF
```

```
                    00000000  01AB  2670          .LONG   ^X00000000
                    0000FFFF  01AF  2671          .LONG   ^X0000FFFF
                    00000000  01B3  2672          .LONG   ^X00000000
                    00FFFFFF  01B7  2673          .LONG   ^X00FFFFFF
                    00000000  01BB  2674          .LONG   ^X00000000
                              01BF  2675
                          04  01BF  2676 5$:      .BYTE   ^X04                    ;STARTING PREFETCH ADDRESS
                          04  01C0  2677          .BYTE   ^X04
                          05  01C1  2678          .BYTE   ^X05
                          05  01C2  2679          .BYTE   ^X05
                          06  01C3  2680          .BYTE   ^X06
                          06  01C4  2681          .BYTE   ^X06
                          07  01C5  2682          .BYTE   ^X07
                          07  01C6  2683          .BYTE   ^X07
                              01C7  2684 6$:
U 11, 7700,8800,5BE4,0310,4870,1812   01C7  2685 ;% 11    PC_R[TEMP4]             ;PC GETS PREFETCH ADDRESS
                              01D2  2689
                    000000FF  01D2  2690 7$:      .LONG   ^X000000FF              ;RESULTS SUBTEST #1
                    FFFFFF00  01D6  2691          .LONG   ^XFFFFFF00
                    000000FF  01DA  2692          .LONG   ^X000000FF
                    FFFFFF00  01DE  2693          .LONG   ^XFFFFFF00
                    000000FF  01E2  2694          .LONG   ^X000000FF
                    FFFFFF00  01E6  2695          .LONG   ^XFFFFFF00
                    000000FF  01EA  2696          .LONG   ^X000000FF
                    FFFFFF00  01EE  2697          .LONG   ^XFFFFFF00
                              01F2  2698 8$:
U 13, 6701,C817,5924,0202,0470,1814   01F2  2699 ;% 13    RDM_WB,ISIZE[LONG],WB_M[XB.PC_PC+I]  ;SOURCE XB1 TO RDM
                              01FD  2703
                    FFFFFF00  01FD  2704 9$:      .LONG   ^XFFFFFF00              ;RESULTS SUBTEST #2
                    000000FF  0201  2705          .LONG   ^X000000FF
                    FFFFFF00  0205  2706          .LONG   ^XFFFFFF00
                    000000FF  0209  2707          .LONG   ^X000000FF
                    FFFFFF00  020D  2708          .LONG   ^XFFFFFF00
                    000000FF  0211  2709          .LONG   ^X000000FF
                    FFFFFF00  0215  2710          .LONG   ^XFFFFFF00
                    000000FF  0219  2711          .LONG   ^X000000FF
                              021D  2712 10$:
U 13, 7300,4800,0364,0300,0470,1814   021D  2713 ;% 13    NOP,STOP.CLOCK          ;DCS ADD 19 SUBTEST #1
                              0228  2717
                              0228  2718 BEGIN_CPU_DATA
                              0002  2719
                              0002  2720 DCS_DATA
                              0001  2721
U 00, 7700,C800,0364,0300,0470,1801   0001  2722 ;% 00    NOP
U 01, 7700,C800,5BE4,0300,6870,1802   000C  2726 ;% 01    MEMSCAR_R[TEMP0]        ;PHY/VIR REG ADD TO MEMSCAR
U 02, 7700,8800,5BE4,0304,6070,1803   0017  2730 ;% 02    MEMSCR_R[TEMP1]         ;SET MME OFF
U 03, 7700,8800,5BE4,0308,6870,1804   0022  2734 ;% 03    MEMSCAR_R[TEMP2]        ;REF CACHE ADD TO MEMSCAR
U 04, 7700,C800,5BE4,030C,6070,1805   002D  2738 ;% 04    MEMSCR_R[TEMP3]         ;TURN CMI OFF
U 05, 7700,0800,5BE4,0310,4A70,1806   0038  2742 ;% 05    VA_R[TEMP4]             ;0'S TO VA REGISTER
U 06, 7700,7800,7FFF,FFFF,8470,1807   0043  2746 ;% 06    LONLIT_[0]              ;TEST PATTERN #1 TO LONGLIT
U 07, 7701,8800,5BE4,02D4,5590,1808   004E  2750 ;% 07    WRITE.LONG R[LONLIT]    ;TEST PATTERN TO CACHE ADD 0
U 08, 7700,4800,5BE4,03D4,5470,1809   0059  2754 ;% 08    WB_R[LONLIT],WCTRL/WDR_WB.UR  ;HOLD DATA FOR EXTRA CYCLE
U 09, 7700,C800,0364,0300,4470,180A   0064  2758 ;% 09    VA_VA+4                 ;INCREMENT VA BY 4
U 0A, 7700,7800,7FFF,FFFF,8470,180B   006F  2762 ;% 0A    LONLIT_[0]              ;TEST PATTERN #2 TO LONGLIT
U 0B, 7701,0800,5BE4,02D4,5590,180C   007A  2766 ;% 0B    WRITE.LONG R[LONLIT]    ;TEST PATTERN TO CACHE ADD 4
```

```
U 0C. 7700,C800,5BE4,03D4,5470,180D  0085  2770 ;% 0C   WB_R[LONLIT],WCTRL/WDR_WB.UR                ;HOLD DATA FOR EXTRA CYCLE
U 0D. 7700,4800,0364,0300,4470,180E  0090  2774 ;% 0D   VA_VA+4                                    ;INCREMENT VA BY 4
U 0E. 7700,F800,7FFF,FFFF,8470,180F  009B  2778 ;% 0E   LONLIT_[0]                                 ;TEST PATTERN #3 TO LONGLIT
U 0F. 7701,8800,5BE4,02D4,5590,1810  00A6  2782 ;% 0F   WRITE.LONG R[LONLIT]                       ;TEST PATTERN TO CACHE ADD 8
U 10. 7700,4800,5BE4,03D4,5470,1811  00B1  2786 ;% 10   WB_R[LONLIT],WCTRL/WDR_WB.UR               ;HOLD DATA FOR EXTRA CYCLE
U 11. 7700,8800,5BE4,0310,4870,1812  00BC  2790 ;% 11   PC_R[TEMP4]                               ;STARTING ADD FOR PREFETCH
U 12. 6701,4817,5924,0202,0470,1813  00C7  2794 ;% 12   RDM_WB,ISIZE[LONG],WB_M[XB.PC_PC+I]      ;READ XB0 TO RDM
U 13. 7300,4800,0364,0300,0470,1814  00D2  2798 ;% 13   NOP,STOP.CLOCK                             ;STOP CPU CLOCK
U 14. 7300,C800,0364,0300,0470,1815  00DD  2802 ;% 14   NOP,STOP.CLOCK                             ;STOP CPU CLOCK
                                     00E8  2806
                                     00E8  2807 RTEMP_DATA
                                     0001  2808
                           00000000  0001  2809          .LONG   ^X00000000                      ;PHY/VIR REG ADDRESS
                           00000000  0005  2810          .LONG   ^X00000000                      ;SET MME OFF
                           0E000000  0009  2811          .LONG   ^X0E000000                      ;REF CACHE ONLY ADDRESS
                           01000000  000D  2812          .LONG   ^X01000000                      ;TURN CMI OFF
                           00000000  0011  2813          .LONG   ^X00000000                      ;ADDRESS FOR VA & PC
                           00000001  0015  2814          .LONG   ^X00000001                      ;ADDRESS FOR PC
                           00000002  0019  2815          .LONG   ^X00000002                      ;ADDRESS FOR PC
                           00000003  001D  2816          .LONG   ^X00000003                      ;ADDRESS FOR PC
                                     0021  2817
                                     0021  2818 END_CPU_DATA
                                     0228  2819
                                     0228  2820 END_TEST_DATA
                                     0228
                                     0228          ;-----------------------------------------------------------------------
                                     0228  2821 ENDTEST
```

```
                        001F        .SBTTL  "TEST   06D     TEST CONSOLE CRAR REGISTER
                        001F  2823 ;*******************************************************************************
                        001F  2824 ;++
                        001F  2825 ;
                        001F  2826 ; FUNCTIONAL DESCRIPTION:
                        001F  2827 ;
                        001F  2828 ;       THIS TEST CHECKS THE DATA INTEGRITY OF THE CONSOLE 'REGISTER ADDRESS'
                        001F  2829 ;       REGISTER (CRAR).
                        001F  2830 ;
                        001F  2831 ;
                        001F  2832 ; TEST ALGORITHM:
                        001F  2833 ;
                        001F  2834 ;       1.  CREATE A TEST LOOP OF 4
                        001F  2835 ;       2.  PUT A TEST PATTERN INTO THE WDREG
                        001F  2836 ;       3.  EXECUTE DCS PROGRAM
                        001F  2837 ;               a.  WRITE THE CRAR WITH THE TEST PATTERN
                        001F  2838 ;               b.  READ THE CRAR TO THE RDM
                        001F  2839 ;       4.  TEST THE CONTENTS OF THE CRAR
                        001F  2840 ;       5.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST LOOPS
                        001F  2841 ;
                        001F  2842 ;
                        001F  2843 ; TEST PATTERNS:
                        001F  2844 ;
                        001F  2845 ;       ITERATION       TEST PATTERN    RESULTS TABLE
                        001F  2846 ;           1             00000000        00000000
                        001F  2847 ;           2             00400000        01000000
                        001F  2848 ;           3             00800000        02000000
                        001F  2849 ;           4             00C00000        03000000
                        001F  2850 ;
                        001F  2851 ;
                        001F  2852 ; LOGIC DESCRIPTION:
                        001F  2853 ;
                        001F  2854 ;       THIS TEST WRITES AND READS TEST PATTERNS TO AND FROM THE CRAR
                        001F  2855 ;       REGISTER WHICH IS LOCATED IN THE CONSLOE CON GATE ARRAY.  ALL
                        001F  2856 ;       FAILURES SHOULD BE CONFINED TO THIS ARRAY OR THE UBI MODULE.
                        001F  2857 ;
                        001F  2858 ;
                        001F  2859 ; ASSUMPTIONS:
                        001F  2860 ;
                        001F  2861 ;       THIS TEST ASSUMES THE WBUS IS OPERATIONAL.
                        001F  2862 ;
                        001F  2863 ;
                        001F  2864 ; ERROR DESCRIPTION:
                        001F  2865 ;
                        001F  2866 ;       EXPECTED CRAR CONTENTS
                        001F  2867 ;       RECEIVED CRAR CONTENTS
                        001F  2868 ;       LOOP COUNT I
                        001F  2869 ;
                        001F  2870 ;--
                        001F  2871 ;*******************************************************************************
                        001F  2872
                        001F  2873
                        001F  2874         INITIALIZE                              ;INIT THE CPU
                        0021  2875         LOOP            I,1,4                   ;CREATE A TEST LOOP OF 4
                        0028  2876         LOADREG         WDREG,1$,I,L            ;TEST PATTERN TO WDREG
```

```
                                0030          ERRLOOP                                   ;ERROR LOOP POINT
                                0032          FETCH          0                          ;START DCS PROGRAM
                                0037          BRSTCLK        3                          ;END DCS PROGRAM
                                003B          CMPREGMSK      WHREG,2$,I,3$,,L,          ;TEST CRAR DATA
                                0047          IFERROR        ,<CON>,<UBI>               ;DATA INCORRECT
                                004E          ENDLOOP        I                          ;END TEST LOOP I
                                0051          SKIP                                      ;GO TO NEXT TEST
                                0058
                                0058
                                0058  BEGIN_TEST_DATA
                                0058
                                0058      ;-----------------------------------------------------------------------
                                0058
                     00000000   0058  1$:      .LONG   ^X00000000                        ;TEST PATTERNS
                     00400000   005C          .LONG   ^X00400000
                     00800000   0060          .LONG   ^X00800000
                     00C00000   0064          .LONG   ^X00C00000
                     00000000   0068  2$:      .LONG   ^X00000000                        ;RESULTS TABLE
                     01000000   006C          .LONG   ^X01000000
                     02000000   0070          .LONG   ^X02000000
                     03000000   0074          .LONG   ^X03000000
                     03000000   0078  3$:      .LONG   ^X03000000                        ;MASK FOR CMPREGMSK PSEUDO OP
                                007C
                                007C
                                007C  BEGIN_CPU_DATA
                                0002
                                0002  DCS_DATA
                                0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 5700,4800,0364,0300,2470,1802  000C  ;% 01   WCTRL/LOADCRAR,WB_RDM                     ;TEST PATTERN TO CRAR
U 02, 6700,4800,0364,0300,3470,1803  0017  ;% 02   RDM_WB,WCTRL/READCRAR                     ;READ CRAR TO RDM
U 03, 7300,C800,0364,0300,0470,1804  0022  ;% 03   NOP,STOP.CLOCK                           ;STOP THE CPU CLOCK
                                002D
                                002D  END_CPU_DATA
                                007C
                                007C
                                007C  END_TEST_DATA
                                007C
                                007C      ;-----------------------------------------------------------------------
                                007C  ENDTEST
```

```
0022          .SBTTL  "TEST   06E     TEST CONSOLE STATUS REGISTERS
0022  2926 ;*******************************************************************************
0022  2927 ;++
0022  2928 ;
0022  2929 ; FUNCTIONAL DESCRIPTION:
0022  2930 ;
0022  2931 ;       THIS TEST CHECKS THE DATA INTEGRITY OF THE FOLLOWING CONSOLE STATUS
0022  2932 ;       REGISTERS:
0022  2933 ;                               1.  XMITTER COMMAND AND STATUS REGISTER
0022  2934 ;                               2.  RECEIVER COMMAND AND STATUS REGISTER
0022  2935 ;                               3.  COMMAND AND STATUS REGISTER
0022  2936 ;
0022  2937 ;
0022  2938 ; TEST ALGORITHM:
0022  2939 ;
0022  2940 ;       1.  CREATE AN INDEX K = 1
0022  2941 ;       2.  CREATE A TEST LOOP OF I = 3
0022  2942 ;       3.  LOAD A STATUS REGISTER ADDRESS INTO THE WDREG
0022  2943 ;       4.  CREATE A SECOND TEST LOOP OF J = 2
0022  2944 ;       5.  LOAD MICRO CODE INTO DCS ACCORDING TO INDEX K
0022  2945 ;       6.  EXECUTE DCS PROGRAM
0022  2946 ;               a.  LOAD THE REGISTER ADDRESS INTO CRAR
0022  2947 ;               b.  WRITE THE TEST PATTERN TO THE REGISTER
0022  2948 ;               c.  READ THE TEST PATTERN BACK TO THE RDM
0022  2949 ;       7.  TEST THE DATA READ BACK
0022  2950 ;       8.  IF NO ERROR INCREMENT INDEX K
0022  2951 ;       9.  GO TO STEP 5 UNTIL LOOP J IS EXHAUSTED
0022  2952 ;      10.  GO TO STEP 3 UNTIL LOOP I IS EXHAUSTED
0022  2953 ;
0022  2954 ;
0022  2955 ; TEST PATTERNS:
0022  2956 ;
0022  2957 ;       ITERATION (K)           TEST RESULTS
0022  2958 ;           1                     00400000
0022  2959 ;           2                     00000000
0022  2960 ;           3                     00400000
0022  2961 ;           4                     00000000
0022  2962 ;           5                     00400000
0022  2963 ;           6                     00800000
0022  2964 ;
0022  2965 ;
0022  2966 ; LOGIC DESCRIPTION:
0022  2967 ;
0022  2968 ;       THIS TEST CHECKS THE STATUS REGISTERS INTERNAL TO THE CONSOLE
0022  2969 ;       CON GATE ARRAY.  ANY ERRORS SHOULD BE CONFINED TO THIS ARRAY OR
0022  2970 ;       THE UBI MODULE.
0022  2971 ;
0022  2972 ;
0022  2973 ; ASSUMPTIONS:
0022  2974 ;
0022  2975 ;       THIS TEST ASSUMES THE CRAR TEST HAS BEEN RUN SUCCESSFULLY
0022  2976 ;
0022  2977 ;
0022  2978 ; ERROR DESCRIPTION:
0022  2979 ;
```

```
                                0022  2980 ;       EXPECTED STATUS REGISTER RESULTS
                                0022  2981 ;       RECEIVED STATUS REGISTER RESULTS
                                0022  2982 ;       LOOP COUNT I
                                0022  2983 ;       LOOP COUNT J
                                0022  2984 ;       LOOP COUNT K
                                0022  2985 ;
                                0022  2986 ;--
                                0022  2987 ;*******************************************************************************
                                0022  2988
                                0022  2989
                                0022  2990         INITIALIZE                                ;INIT THE CPU
                                0024  2991         LDINDEX         K,1                       ;MAKE INDEX K = 1
                                002B  2992         LOOP            I,1,3                     ;CREATE A TEST LOOP OF 3
                                0032  2993         LOADREG         WDREG,1$,I,L              ;LOAD REG ADDRESS INTO WDREG
                                003A  2994         LOOP            J,1,2                     ;CREATE A LOOP OF 2
                                0041  2995         LOADDCS         4$,K,02,1                 ;MODIFY MICRO CODE FOR LOOP
                                0048  2996         ERRLOOP                                   ;ERROR LOOP POINT
                                004A  2997         FETCH           0                         ;START DCS PROGRAM
                                004F  2998         BRSTCLK         4                         ;END DCS PROGRAM
                                0053  2999         CMPREGMSK       WHREG,2$,J,3$,I,L         ;TEST REGISTER CONTENTS
                                005F  3000         IFERROR         ,<CON>,<UBI>              ;INCORRECT DATA
                                0066  3001         INCINDEX        K                         ;INCREMENT INDEX K BY 1
                                0069  3002         ENDLOOP         J                         ;END TEST LOOP J
                                006C  3003         ENDLOOP         I                         ;END TEST LOOP I
                                006F  3004         SKIP                                      ;GO TO NEXT TEST
                                0076  3005
                                0076  3006
                                0076  3007 BEGIN_TEST_DATA
                                0076
                                0076       ;-------------------------------------------------------------------------
                                0076  3008
                    00400000    0076  3009 1$:      .LONG    ^X00400000                      ;REGISTER ADDRESSES
                    00800000    007A  3010          .LONG    ^X00800000
                    00C00000    007E  3011          .LONG    ^X00C00000
                    00400000    0082  3012 2$:      .LONG    ^X00400000                      ;RESULTS TABLE
                    00800000    0086  3013          .LONG    ^X00800000
                    00400000    008A  3014 3$:      .LONG    ^X00400000                      ;MASK FOR CMPREGMSK PSEUDO OP
                    00400000    008E  3015          .LONG    ^X00400000
                    00C00000    0092  3016          .LONG    ^X00C00000
                                0096  3017 4$:                                              ;MICRO CODE FOR EACH LOOP
U 02, 7700,4800,5BE4,0300,2C70,1803 0096  3018 ;% 02    WCTRL/CONWRITE,WB_R[TEMP0]          ;WRITE REGISTER WITH TEMP0
U 02, 7700,0800,5BE4,0304,2C70,1803 00A1  3022 ;% 02    WCTRL/CONWRITE,WB_R[TEMP1]          ;WRITE REGISTER WITH TEMP1
U 02, 7700,4800,5BE4,0300,2C70,1803 00AC  3026 ;% 02    WCTRL/CONWRITE,WB_R[TEMP0]          ;WRITE REGISTER WITH TEMP0
U 02, 7700,0800,5BE4,0304,2C70,1803 00B7  3030 ;% 02    WCTRL/CONWRITE,WB_R[TEMP1]          ;WRITE REGISTER WITH TEMP1
U 02, 7700,0800,5BE4,0308,2C70,1803 00C2  3034 ;% 02    WCTRL/CONWRITE,WB_R[TEMP2]          ;WRITE REGISTER WITH TEMP2
U 02, 7700,4800,5BE4,030C,2C70,1803 00CD  3038 ;% 02    WCTRL/CONWRITE,WB_R[TEMP3]          ;WRITE REGISTER WITH TEMP1
                                00D8  3042
                                00D8  3043 BEGIN_CPU_DATA
                                0002  3044
                                0002  3045 DCS_DATA
                                0001  3046
U 00, 7700,C800,0364,0300,0470,1801 0001  3047 ;% 00    NOP
U 01, 5700,4800,0364,0300,2470,1802 000C  3051 ;% 01    WCTRL/LOADCRAR,WB_RDM               ;WRITE THE REGISTER ADDRESS
U 02, 7700,4800,5BE4,0300,2C70,1803 0017  3055 ;% 02    WCTRL/CONWRITE,WB_R[TEMP0]          ;WRITE DATA TO REGISTER
U 03, 6700,4800,0364,0300,3C70,1804 0022  3059 ;% 03    RDM_WB,WCTRL/CONREAD                ;READ THE REGISTER TO THE RDM
```

```
U 04, 7300,4800,0364,0300,0470,1805  002D  3063 ;% 04   NOP,STOP.CLOCK                          ;STOP THE CLOCK
                                     0038  3067
                                     0038  3068 RTEMP_DATA
                                     0001  3069
                           00400000  0001  3070          .LONG   ^X00400000                   ;TEST DATA
                           00800000  0005  3071          .LONG   ^X00800000
                           00100000  0009  3072          .LONG   ^X00100000
                           00C00000  000D  3073          .LONG   ^X00C00000
                                     0011  3074
                                     0011  3075 END_CPU_DATA
                                     00D8  3076
                                     00D8  3077
                                     00D8  3078 END_TEST_DATA
                                     00D8
                                     00D8       ;-------------------------------------------------------------------------
                                     00D8  3079 ENDTEST
```

```
                        001C        .SBTTL  "TEST   06F     TEST TU58 TRAR REGISTER
                        001C  3081 ;*******************************************************************************
                        001C  3082 ;++
                        001C  3083 ;
                        001C  3084 ; FUNCTIONAL DESCRIPTION:
                        001C  3085 ;
                        001C  3086 ;       THIS TEST CHECKS THE DATA INTEGRITY OF THE TU58 TRAR REGISTER
                        001C  3087 ;
                        001C  3088 ;
                        001C  3089 ; TEST ALGORITHM:
                        001C  3090 ;
                        001C  3091 ;       1.  CREATE A TEST LOOP OF I = 4
                        001C  3092 ;       2.  PUT A TEST PATTERN INTO THE WDREG
                        001C  3093 ;       3.  EXECUTE DCS PROGRAM
                        001C  3094 ;               a.  WRITE TEST PATTERN TO THE TRAR REGISTER
                        001C  3095 ;               b.  READ THE TEST PATTERN BACK TO THE RDM
                        001C  3096 ;               c.  STOP THE CPU CLOCK
                        001C  3097 ;       4.  TEST THE RETURNED DATA
                        001C  3098 ;       5.  IF NO ERROR GO TO STEP 2 UNTIL LOOP I IS EXHAUSTED
                        001C  3099 ;
                        001C  3100 ;
                        001C  3101 ; TEST PATTERNS:
                        001C  3102 ;
                        001C  3103 ;       ITERATION       RESULTS
                        001C  3104 ;           1           00000000
                        001C  3105 ;           2           01000000
                        001C  3106 ;           3           02000000
                        001C  3107 ;           4           03000000
                        001C  3108 ;
                        001C  3109 ;
                        001C  3110 ; LOGIC DESCRIPTION:
                        001C  3111 ;
                        001C  3112 ;       THIS TEST CHECKS THE TU58 TRAR REGISTER WHICH IS LOCATED INTERNAL
                        001C  3113 ;       TO THE CON GATE ARRAY.  ALL FAILURES SHOULD BE CONFINED TO THIS ARRAY.
                        001C  3114 ;
                        001C  3115 ;
                        001C  3116 ; ASSUMPTIONS:
                        001C  3117 ;
                        001C  3118 ;       THIS TEST ASSUMES THE WBUS IS OPERATIONAL.
                        001C  3119 ;
                        001C  3120 ;
                        001C  3121 ; ERROR DESCRIPTION:
                        001C  3122 ;
                        001C  3123 ;       EXPECTED TRAR REGISTER CONTENTS
                        001C  3124 ;       RECEIVED TRAR REGISTER CONTENTS
                        001C  3125 ;       LOOP COUNT I
                        001C  3126 ;
                        001C  3127 ;--
                        001C  3128 ;*******************************************************************************
                        001C  3129
                        001C  3130
                        001C  3131         INITIALIZE                              ;INIT THE CPU
                        001E  3132         LOOP            I,1,4                   ;CREATE A TEST LOOP OF 4
                        0025  3133         LOADREG         WDREG,1$,I,L            ;TEST PATTERN TO WDREG
                        002D  3134         ERRLOOP                                 ;ERROR LOOP POINT
```

```
                          002F  3135          FETCH           0                       ;START DCS PROGRAM
                          0034  3136          BRSTCLK         3                       ;END DCS PROGRAM
                          0038  3137          CMPREGMSK       WHREG,2$,I,3$,,L,       ;TEST CRAR DATA
                          0044  3138          IFERROR         ,<CON>,<UBI>            ;DATA INCORRECT
                          004B  3139          ENDLOOP         I                       ;END TEST LOOP I
                          004E  3140          SKIP                                    ;GO TO NEXT TEST
                          0055  3141
                          0055  3142
                          0055  3143 BEGIN_TEST_DATA
                          0055
                          0055          ;-----------------------------------------------------------------------
                          0055  3144
                 00000000 0055  3145 1$:      .LONG    ^X00000000                     ;TEST PATTERNS
                 00400000 0059  3146          .LONG    ^X00400000
                 00800000 005D  3147          .LONG    ^X00800000
                 00C00000 0061  3148          .LONG    ^X00C00000
                 00000000 0065  3149 2$:      .LONG    ^X00000000                     ;RESULTS TABLE
                 01000000 0069  3150          .LONG    ^X01000000
                 02000000 006D  3151          .LONG    ^X02000000
                 03000000 0071  3152          .LONG    ^X03000000
                 03000000 0075  3153 3$:      .LONG    ^X03000000                     ;MASK FOR CMPREGMSK PSEUDO OP
                          0079  3154
                          0079  3155
                          0079  3156 BEGIN_CPU_DATA
                          0002  3157
                          0002  3158 DCS_DATA
                          0001  3159
U 00, 7700,C800,0364,0300,0470,1801  0001  3160 ;% 00    NOP
U 01, 5700,C800,0364,0300,2670,1802  000C  3164 ;% 01    WCTRL/LOADTRAR,WB_RDM          ;TEST PATTERN TO TRAR
U 02, 6700,C800,0364,0300,3670,1803  0017  3168 ;% 02    RDM_WB,WCTRL/READTRAR          ;READ TRAR TO RDM
U 03, 7300,C800,0364,0300,0470,1804  0022  3172 ;% 03    NOP,STOP.CLOCK                 ;STOP THE CPU CLOCK
                          002D  3176
                          002D  3177 END_CPU_DATA
                          0079  3178
                          0079  3179
                          0079  3180 END_TEST_DATA
                          0079
                          0079          ;-----------------------------------------------------------------------
                          0079  3181 ENDTEST
```

```
                        001F        .SBTTL  "TEST   070     TEST TU58 STATUS REGISTERS
                        001F  3183 ;*******************************************************************************
                        001F  3184 ;++
                        001F  3185 ;
                        001F  3186 ; FUNCTIONAL DESCRIPTION:
                        001F  3187 ;
                        001F  3188 ;       THIS TEST CHECKS THE DATA INTEGRITY OF THE FOLLOWING TU58 CON
                        001F  3189 ;       GATE ARRAY REGISTERS:
                        001F  3190 ;               1.  XMITTER COMMAND AND STATUS REGISTER
                        001F  3191 ;               2.  RECEIVER COMMAND AND STATUS REGISTER
                        001F  3192 ;               3.  COMMAND AND STATUS REGISTER
                        001F  3193 ;
                        001F  3194 ;
                        001F  3195 ; TEST ALGORITHM:
                        001F  3196 ;
                        001F  3197 ;       1.  CREATE A TEST LOOP I = 3
                        001F  3198 ;       2.  LOAD A REGISTER ADDRESS INTO THE WDREG
                        001F  3199 ;       3.  CREATE A SECOND TEST LOOP J = 2
                        001F  3200 ;       4.  LOAD MICRO CODE INTO DCS ACCORDING TO J
                        001F  3201 ;       5.  EXECUTE DCS PROGRAM
                        001F  3202 ;               a.  LOAD REGISTER ADDRESS INTO TRAR
                        001F  3203 ;               b.  WRITE A TEST PATTERN TO THE SELECTED REGISTER
                        001F  3204 ;               c.  READ THE TEST PATTERN BACK TO THE RDM
                        001F  3205 ;               d.  STOP THE CPU CLOCK
                        001F  3206 ;       6.  TEST FOR THE CORRECT PATTERN
                        001F  3207 ;       7.  IF NO ERROR GO TO STEP 4 UNTIL LOOP J IS EXHAUSTED
                        001F  3208 ;       8.  GO TO STEP 2 UNTIL LOOP I IS EXHAUSTED
                        001F  3209 ;
                        001F  3210 ;
                        001F  3211 ; TEST PATTERNS:
                        001F  3212 ;
                        001F  3213 ;       ITERATION (I)   ITERATION (J)   RESULTS
                        001F  3214 ;           1               1           00400000
                        001F  3215 ;           1               2           00000000
                        001F  3216 ;           2               1           00400000
                        001F  3217 ;           2               2           00000000
                        001F  3218 ;           3               1           00000000
                        001F  3219 ;           3               2           00800000
                        001F  3220 ;
                        001F  3221 ;
                        001F  3222 ; LOGIC DESCRIPTION:
                        001F  3223 ;
                        001F  3224 ;       THIS TEST WRITES AND READS THE TU58 CON GATE ARRAY STATUS REGISTERS.
                        001F  3225 ;       ALL FAILURES SHOULD BE WITHIN THE ARRAY.
                        001F  3226 ;
                        001F  3227 ;
                        001F  3228 ; ASSUMPTIONS:
                        001F  3229 ;
                        001F  3230 ;       THIS TEST ASSUMES THE WBUS IS OPERATIONAL.
                        001F  3231 ;
                        001F  3232 ;
                        001F  3233 ; ERROR DESCRIPTION:
                        001F  3234 ;
                        001F  3235 ;       EXPECTED REGISTER CONTENTS
                        001F  3236 ;       RECEIVED REGISTER CONTENTS
```

```
                                001F  ;       LOOP COUNT I
                                001F  ;       LOOP COUNT J
                                001F  ;
                                001F  ;--
                                001F  ;*******************************************************************************
                                001F
                                001F
                                001F          INITIALIZE                                  ;INIT THE CPU
                                0021          LOOP            I,1,3                       ;CREATE A TEST LOOP OF 3
                                0028          LOADREG         WDREG,1$,I,L                ;LOAD REG ADDRESS INTO WDREG
                                0030          LOOP            J,1,2                       ;CREATE A LOOP OF 2
                                0037          LOADDCS         4$,J,02,1                   ;MODIFY MICRO CODE FOR LOOP
                                003E          ERRLOOP                                     ;ERROR LOOP POINT
                                0040          FETCH           0                           ;START DCS PROGRAM
                                0045          BRSTCLK         4                           ;END DCS PROGRAM
                                0049          CMPREGMSK       WHREG,2$,J,3$,I,L           ;TEST REGISTER CONTENTS
                                0055          IFERROR         ,<CON>,<UBI>                ;INCORRECT DATA
                                005C          ENDLOOP         J                           ;END TEST LOOP J
                                005F          ENDLOOP         I                           ;END TEST LOOP I
                                0062          SKIP                                        ;GO TO NEXT TEST
                                0069
                                0069
                                0069  BEGIN_TEST_DATA
                                0069
                                0069  ;-----------------------------------------------------------------------
                                0069
                     00400000   0069  1$:     .LONG   ^X00400000                          ;REGISTER ADDRESSES
                     00800000   006D          .LONG   ^X00800000
                     00C00000   0071          .LONG   ^X00C00000
                     00400000   0075  2$:     .LONG   ^X00400000                          ;RESULTS TABLE
                     00800000   0079          .LONG   ^X00800000
                     00400000   007D  3$:     .LONG   ^X00400000                          ;MASK FOR CMPREGMSK PSEUDO OP
                     00400000   0081          .LONG   ^X00400000
                     00800000   0085          .LONG   ^X00800000
                                0089  4$:                                                 ;MICRO CODE FOR EACH LOOP
U 02, 7700,C800,5BE4,0300,2E70,1803  0089  ;% 02   WCTRL/TU58WRITE,WB_R[TEMP0]                ;WRITE REGISTER WITH TEMP0
U 02, 7700,8800,5BE4,0304,2E70,1803  0094  ;% 02   WCTRL/TU58WRITE,WB_R[TEMP1]                ;WRITE REGISTER WITH TEMP1
                                009F
                                009F
                                009F  BEGIN_CPU_DATA
                                0002
                                0002  DCS_DATA
                                0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 5700,C800,0364,0300,2670,1802  000C  ;% 01   WCTRL/LOADTRAR,WB_RDM                      ;WRITE THE REGISTER ADDRESS
U 02, 7700,C800,5BE4,0300,2E70,1803  0017  ;% 02   WCTRL/TU58WRITE,WB_R[TEMP0]                ;WRITE DATA TO REGISTER
U 03, 6700,C800,0364,0300,3E70,1804  0022  ;% 03   RDM_WB,WCTRL/TU58READ                      ;READ THE REGISTER TO THE RDM
U 04, 7300,4800,0364,0300,0470,1805  002D  ;% 04   NOP,STOP.CLOCK                             ;STOP THE CLOCK
                                0038
                                0038  RTEMP_DATA
                                0001
                     00400000   0001          .LONG   ^X00400000                          ;TEST DATA
                     00800000   0005          .LONG   ^X00800000
                                0009
                                0009  END_CPU_DATA
```

```
                        009F  3311
                        009F  3312
                        009F  3313 END_TEST_DATA
                        009F
                        009F       ;-----------------------------------------------------------------------
                        009F  3314 ENDTEST
```

```
                        001E        .SBTTL  "TEST   071     TEST IR AND OSR REGISTERS
                        001E  3316 ;*******************************************************************************
                        001E  3317 ;++
                        001E  3318 ;
                        001E  3319 ; FUNCTIONAL DESCRIPTION:
                        001E  3320 ;
                        001E  3321 ;       THIS TEST WILL CHECK THE DATA INTEGRITY OF THE INSTRUCTION REGISTER
                        001E  3322 ;       (IR) AND OPERAND SPECIFIER REGISTER (OSR) BY WRITING AND READING TEST
                        001E  3323 ;       PATTERNS TO BOTH REGISTERS.  THE REGISTERS WILL BE WRITTEN USING THE
                        001E  3324 ;       BUT FIELD MICRO-ORDER "IRD1TST".  THEY WILL BE READ BACK TO THE MBUS
                        001E  3325 ;       USING "WCTRL/MDR_IR" AND "CCPSL/MDR_OSR.CCBR_BRATST".
                        001E  3326 ;
                        001E  3327 ;       NOTE:   IF THIS TEST FAILS AND THE MIC MODULE IS NOT KNOWN GOOD
                        001E  3328 ;               THEN YOU SHOULD RUN THE MIC TESTS AT THIS TIME TO VERIFY
                        001E  3329 ;               IT IS OPERATIONAL.
                        001E  3330 ;
                        001E  3331 ;
                        001E  3332 ; TEST ALGORITHM:
                        001E  3333 ;
                        001E  3334 ;       SUBTEST #1 & 2:
                        001E  3335 ;
                        001E  3336 ;       1.  MODIFY DCS MICRO WORDS FOR THIS SUBTEST
                        001E  3337 ;       2.  CREATE A TEST LOOP
                        001E  3338 ;       3.  LOAD LONLIT WITH TEST PATTERN
                        001E  3339 ;       4.  EXECUTE DCS PROGRAM
                        001E  3340 ;               1.  DISABLE ^P INTERRUPTS
                        001E  3341 ;               2.  SET PSL TO NATIVE MODE AND IPL = 1F
                        001E  3342 ;               3.  LOAD 0'S INTO VIRTUAL ADDRESS REGISTER
                        001E  3343 ;               4.  WRITE TEST PATTERN INTO LONGLIT REGISTER
                        001E  3344 ;               5.  MOVE TEST PATTERN FROM LONLGLIT REGISTER TO CACHE
                        001E  3345 ;                   ADDRESS 0
                        001E  3346 ;               6.  LOAD 0'S INTO PROGRAM COUNTER
                        001E  3347 ;               7.  MOVE TEST PATTERN INTO IR AND OSR REGISTERS
                        001E  3348 ;               8.  READ IR OR OSR REGISTER TO MDR REGISTER
                        001E  3349 ;               9.  READ MDR REGISTER TO RDM
                        001E  3350 ;       5.  TEST VBUS FOR LD OSR (SUBTEST 2 ONLY)
                        001E  3351 ;       6.  COMPARE RESULTS (EXPECTED AND RECEIVED)
                        001E  3352 ;       7.  IF NO ERROR GO TO STEP 3 FOR REMAINING TEST PATTERNS
                        001E  3353 ;
                        001E  3354 ;       SUBTEST #3 & 4:
                        001E  3355 ;
                        001E  3356 ;       1.  CHANGE DCS ADDRESS 03 TO SPECIFY COMPATABILITY MODE
                        001E  3357 ;       2.  EXECUTE SAME DCS PROGRAM AS IN SUBTEST #1 & 2 WITH ONLY
                        001E  3358 ;           ONE PASS USING TEST PATTERN #5
                        001E  3359 ;
                        001E  3360 ;
                        001E  3361 ;
                        001E  3362 ; TEST PATTERNS:
                        001E  3363 ;
                        001E  3364 ;       ITERATION       TEST PATTERN
                        001E  3365 ;           1               AAAA
                        001E  3366 ;           2               5555
                        001E  3367 ;           3               3333
                        001E  3368 ;           4               0F0F
                        001E  3369 ;           5               00FF
```

```
                        001E  3370 ;
                        001E  3371 ;
                        001E  3372 ; LOGIC DESCRIPTION:
                        001E  3373 ;
                        001E  3374 ;       THIS TEST CHECKS THE LOADING AND READING OF THE INSTRUCTION REGISTER
                        001E  3375 ;       AND OPERAND SPECIFIER REGISTER.  BOTH OF THESE REGISTERS ARE LOCATED
                        001E  3376 ;       IN THE "IRD" GATE ARRAY ON THE DPM MODULE.  THIS SHOULD BE THE MOST
                        001E  3377 ;       LIKELY ARRAY TO FAIL.  OTHER GATE ARRAYS TO WATCH ARE:  "MDR" WHICH
                        001E  3378 ;       CONTAINS THE DATA PATH TO AND FROM THE IRD; "ADK" WHICH CONTROLS
                        001E  3379 ;       THE DBUS SELECT WITHIN THE MDR; AND "CAK" WHICH CONTROLS THE DBUS
                        001E  3380 ;       ROTATOR.  IF REPLACING THESE ARRAYS DOES NOT FIX YOUR PROBLEM THEN
                        001E  3381 ;       YOUR ONLY HOPE IS PRAYER.
                        001E  3382 ;       TO HELP ISOLATE ANY PROBLEMS THE LD OSR SIGNAL IS TESTED ON THE
                        001E  3383 ;       VBUS IN SUBTEST 2.
                        001E  3384 ;
                        001E  3385 ; ASSUMPTIONS:
                        001E  3386 ;
                        001E  3387 ;       THIS TEST ASSUMES THE WBUS, CACHE, VA, PC, AND MBUS ARE OPERATIONAL.
                        001E  3388 ;
                        001E  3389 ;
                        001E  3390 ; ERROR DESCRIPTION:
                        001E  3391 ;
                        001E  3392 ;       SUBTEST 1 ONLY:
                        001E  3393 ;               FIRST IFERROR:
                        001E  3394 ;                       EXPECTED CS HALT ADDRESS
                        001E  3395 ;                       RECEIVED CS HALT ADDRESS
                        001E  3396 ;                       LOOP COUNT I
                        001E  3397 ;
                        001E  3398 ;       SUBTEST 2 ONLY:
                        001E  3399 ;               FIRST IFERROR:
                        001E  3400 ;                       EXPECTED VBUS DATA (LD OSR)
                        001E  3401 ;                       RECEIVED VBUS DATA
                        001E  3402 ;                       LOOP COUNT I
                        001E  3403 ;
                        001E  3404 ;               SECOND IFERROR:
                        001E  3405 ;                       EXPECTED OSR DATA
                        001E  3406 ;                       RECEIVED OSR DATA
                        001E  3407 ;                       LOOP COUNT I
                        001E  3408 ;
                        001E  3409 ;       ALL OTHER SUBTESTS:
                        001E  3410 ;               EXPECTED IR/OSR DATA
                        001E  3411 ;               RECEIVED IR/OSR DATA
                        001E  3412 ;               LOOP COUNT I (SUBTEST 1)
                        001E  3413 ;
                        001E  3414 ;--
                        001E  3415 ;*******************************************************************************
                        001E  3416
                        001E  3417         SUBTEST                                         ;SUBTEST #1
                        0021
                        0021       ;////////////////////////////////////////////////////////////////////////////////
                        0021  3418 ;+
                        0021  3419 ;SUBTEST #1
                        0021  3420 ;
                        0021  3421 ;       TEST INSTRUCTION REGISTER
                        0021  3422 ;-
```

```
                        0021  3423          INITIALIZE                                ;INIT CPU
                        0023  3424          EXPUTRAP                                  ;EXPECT POSSIBLE MICRO TRAPS
                        0025  3425          LOADDCS         10$,,03,1                 ;SET NATIVE MODE MICRO WORD
                        002C  3426          LOOP            I,1,5                     ;CREATE A TEST LOOP
                        0033  3427          LDFIELD         2$,I,W,1$,31,62,LON       ;TEST PATTERN TO MICRO WORD
                        0044  3428          LOADDCS         1$,,05,1                  ;TEST PATTERN TO DCS
                        004B  3429          ERRLOOP                                   ;ERROR LOOP
                        004D  3430          FETCH           0                         ;START DCS PROGRAM
                        0052  3431          BRSTCLK         <^X0C>                    ;END DCS PROGRAM
                        0056  3432          CMPREGMSK       CSTRP,12$,,13$,,W,        ;TEST FOR LAST DCS ADDRESS
                        0062  3433          IFERROR         ,,<<UBI><FPA>>            ;UNEXPECTED INTERRUPT
                        0069  3434          CMPREG          WHREG,3$,I,L,             ;COMPARE RESULTS
                        0072  3435          IFERROR         ,<<IRD><MDR><ADK><CAK>>,<MIC>
                        007C  3436          ENDLOOP         I                         ;END TEST LOOP
                        007F  3437
                        007F  3438          SUBTEST                                   ;SUBTEST #2
                        0082
                        0082              ;////////////////////////////////////////////////////////////////////////////////
                        0082  3439  ;+
                        0082  3440  ;SUBTEST #2
                        0082  3441  ;
                        0082  3442  ;       TEST OPERAND SPECIFIER REGISTER
                        0082  3443  ;-
                        0082  3444          INITIALIZE                                ;INIT CPU
                        0084  3445          LOADDCS         4$,,<^X0A>,1              ;READ OSR MICRO WORD
                        008B  3446          LOOP            I,1,5                     ;CREATE A TEST LOOP
                        0092  3447          LDFIELD         2$,I,W,1$,31,62,LON       ;TEST PATTERN TO MICRO WORD
                        00A3  3448          LOADDCS         1$,,05,1                  ;TEST PATTERN TO DCS
                        00AA  3449          ERRLOOP                                   ;ERROR LOOP
                        00AC  3450          FETCH           0                         ;START DCS PROGRAM
                        00B1  3451          BRSTCLK         <^X0C>                    ;END DCS PROGRAM
                        00B5  3452          CMPVBUS         11$                       ;TEST LD OSR SIGNAL
                        00BA  3453          IFERROR         ,<MSQ>,                   ;SIGNAL NOT PRESENT
                        00C0  3454          CMPREG          WHREG,5$,I,L,             ;COMPARE RESULTS
                        00C9  3455          IFERROR         ,<<IRD><MDR><ADK><CAK>>,<MIC>
                        00D3  3456          ENDLOOP         I                         ;END TEST LOOP
                        00D6  3457
                        00D6  3458          SUBTEST                                   ;SUBTEST #3
                        00D9
                        00D9              ;////////////////////////////////////////////////////////////////////////////////
                        00D9  3459  ;+
                        00D9  3460  ;SUBTEST #3
                        00D9  3461  ;
                        00D9  3462  ;       TEST OPERAND SPECIFIER REGISTER IN COMPATABILITY MODE
                        00D9  3463  ;-
                        00D9  3464          INITIALIZE                                ;INIT CPU
                        00DB  3465          LOADDCS         8$,,03,1                  ;SET CM BIT MICRO WORD
                        00E2  3466          ERRLOOP                                   ;ERROR LOOP
                        00E4  3467          FETCH           0                         ;START DCS PROGRAM
                        00E9  3468          BRSTCLK         <^X0C>                    ;END DCS PROGRAM
                        00ED  3469          CMPREG          WHREG,6$,,L,              ;COMPARE RESULTS
                        00F6  3470          IFERROR         ,<<IRD><MDR><ADK><CAK>>,<MIC>
                        0100  3471
                        0100  3472          SUBTEST                                   ;SUBTEST #4
                        0103
```

```
                                  0103        ;////////////////////////////////////////////////////////////////////////////
                                  0103  3473  ;+
                                  0103  3474  ;SUBTEST #4
                                  0103  3475  ;
                                  0103  3476  ;       TEST INSTRUCTION REGISTER IN COMPATABILITY MODE
                                  0103  3477  ;-
                                  0103  3478          INITIALIZE                                  ;INIT CPU
                                  0105  3479          LOADDCS         9$,,<^X0A>,1                ;READ IR MICRO WORD
                                  010C  3480          ERRLOOP                                     ;ERROR LOOP
                                  010E  3481          FETCH           0                           ;START DCS PROGRAM
                                  0113  3482          BRSTCLK         <^X0C>                      ;END DCS PROGRAM
                                  0117  3483          CMPREG          WHREG,7$,,L,                ;COMPARE RESULTS
                                  0120  3484          IFERROR         ,<<IRD><MDR><ADK><CAK>>,<MIC>
                                  012A  3485          SKIP                                        ;GO TO NEXT TEST
                                  0131  3486
                                  0131  3487  BEGIN_TEST_DATA
                                  0131
                                  0131        ;-----------------------------------------------------------------------
                                  0131  3488
                                  0131  3489  1$:
U 05, 7700,F800,7FFF,FFFF,8470,1806  0131  3490  ;% 05   LONLIT_[0]                                 ;TEST PATTERN TO LONGLIT
                                  013C  3494
                             AAAA  013C  3495  2$:     .WORD   ^XAAAA                              ;TEST PATTERNS
                             5555  013E  3496          .WORD   ^X5555
                             3333  0140  3497          .WORD   ^X3333
                             0F0F  0142  3498          .WORD   ^X0F0F
                             00FF  0144  3499          .WORD   ^X00FF
                                  0146  3500
                         000000AA  0146  3501  3$:     .LONG   ^X000000AA                          ;RESULTS SUBTEST #1
                         00000055  014A  3502          .LONG   ^X00000055
                         00000033  014E  3503          .LONG   ^X00000033
                         0000000F  0152  3504          .LONG   ^X0000000F
                         000000FF  0156  3505          .LONG   ^X000000FF
                                  015A  3506  4$:
U 0A, 7700,C800,0364,0300,5E70,180B  015A  3507  ;% 0A   CCPSL/MDR_OSR.CCBR_BRATST                   ;READ OSR TO MDR REG
                                  0165  3511
                         000000AA  0165  3512  5$:     .LONG   ^X000000AA                          ;RESULTS SUBTEST #2
                         00000055  0169  3513          .LONG   ^X00000055
                         00000033  016D  3514          .LONG   ^X00000033
                         0000000F  0171  3515          .LONG   ^X0000000F
                         00000000  0175  3516          .LONG   ^X00000000
                                  0179  3517
                         000000FF  0179  3518  6$:     .LONG   ^X000000FF                          ;RESULTS SUBTEST #3
                                  017D  3519
                         00000000  017D  3520  7$:     .LONG   ^X00000000                          ;RESULTS SUBTEST #4
                                  0181  3521
                                  0181  3522  8$:
U 03, 7700,0800,5BF4,030C,0070,1804  0181  3523  ;% 03   PSL_R[TEMP3]                                ;SET CM BIT IN PSL
                                  018C  3527
                                  018C  3528  9$:
U 0A, 7700,4800,0364,0300,5670,180B  018C  3529  ;% 0A   WCTRL/MDR_IR                                ;READ IR TO MDR REGISTER
                                  0197  3533
                                  0197  3534  10$:
U 03, 7700,4800,5BF4,0308,0070,1804  0197  3535  ;% 03   PSL_R[TEMP2]                                ;SET NATIVE MODE IN PSL
                                  01A2  3539
```

```
                          01  01A2  3540 11$:    .BYTE   ^X1                          ;LD OSR SIGNAL
                              01A3  3541         VBUSDATA        <^X1E>,0
                        180C  01A4  3542 12$:    .WORD   ^X180C                       ;LAST DCS ADDRESS
                        3FFF  01A6  3543 13$:    .WORD   ^X3FFF                       ;MASK FOR CMPREGMSK PSEUDO OP
                              01A8  3544
                              01A8  3545 BEGIN_CPU_DATA
                              0002  3546
                              0002  3547 DCS_DATA
                              0001  3548
U 00, 7700,C800,0364,0300,0470,1801  0001  3549 ;% 00   NOP
U 01, 7700,4800,5BE4,0300,2470,1802  000C  3553 ;% 01   WCTRL/LOADCRAR,WB_R[TEMP0]        ;DISABLE ^P INTERRUPT
U 02, 7700,0800,5BE4,0304,2C70,1803  0017  3557 ;% 02   WCTRL/CONWRITE,WB_R[TEMP1]        ;...
U 03, 7700,4800,5BF4,0308,0070,1804  0022  3561 ;% 03   PSL_R[TEMP2]                       ;SET NATIVE MODE
U 04, 7700,4800,5BE4,03D8,4A70,1805  002D  3565 ;% 04   VA_R[ZERO]                         ;0'S TO VA REGIATER
U 05, 7700,F800,7FFF,FFFF,8470,1806  0038  3569 ;% 05   LONLIT_[0]                         ;TEST PATTERN TO LONGLIT REG
U 06, 7701,8800,5BE4,02D4,5590,1807  0043  3573 ;% 06   WRITE.LONG R[LONLIT]               ;TEST PATTERN TO CACHE ADD 0
U 07, 7700,C800,5BE4,03D4,5470,1808  004E  3577 ;% 07   WB_R[LONLIT],WCTRL/WDR_WB.UR      ;HOLD DATA FOR EXTRA CYCLE
U 08, 7700,C800,5BE4,03D8,4870,1809  0059  3581 ;% 08   PC_R[ZERO]                         ;0'S TO PROGRAM COUNTER
U 09, 7600,4800,0364,1700,0470,180A  0064  3585 ;% 09   BUT/IRD1TST,STROBE.V.BUS           ;LOAD IR AND OSR REGISTERS
U 0A, 7700,4800,0364,0300,5670,180B  006F  3589 ;% 0A   WCTRL/MDR_IR                       ;READ IR TO MDR REGISTER
U 0B, 6700,0812,5924,0300,0470,180C  007A  3593 ;% 0B   RDM_WB,WB_M[MDR]                   ;RDM GETS MDR REGISTER
U 0C, 7300,C800,0364,0300,0470,180D  0085  3597 ;% 0C   NOP,STOP.CLOCK                     ;STOP CPU CLOCK
                              0090  3601
                              0090  3602 RTEMP_DATA
                              0001  3603
                    00C00000  0001  3604         .LONG   ^X00C00000                   ;DISABLE ^P INTERRUPT
                    00400000  0005  3605         .LONG   ^X00400000                   ;...
                    001F0000  0009  3606         .LONG   ^X001F0000                   ;SET NATIVE MODE
                    801F0000  000D  3607         .LONG   ^X801F0000                   ;SET CM BIT IN PSL
                              0011  3608
                              0011  3609 END_CPU_DATA
                              01A8  3610
                              01A8  3611 END_TEST_DATA
                              01A8
                              01A8       ;-----------------------------------------------------------------------
                              01A8  3612 ENDTEST
```

```
                     0022        .SBTTL  "TEST   072     TEST INSTRUCTION FETCH SIGNAL
                     0022  3614 ;*******************************************************************************
                     0022  3615 ;++
                     0022  3616 ;
                     0022  3617 ; FUNCTIONAL DESCRIPTION:
                     0022  3618 ;
                     0022  3619 ;       THIS TEST WILL VERIFY THAT THE INSTRUCTION FETCH SIGNAL IS PRESENT
                     0022  3620 ;       ON IRD1TEST.
                     0022  3621 ;
                     0022  3622 ;
                     0022  3623 ; TEST ALGORITHM:
                     0022  3624 ;
                     0022  3625 ;       1.  EXECUTE DCS PROGRAM
                     0022  3626 ;               a.  LOAD THE PC REGISTER WITH ZERO TO CAUSE PREFETCH
                     0022  3627 ;               b.  PERFORM AN IRD1TEST
                     0022  3628 ;       2.  TEST THE VBUS FOR INSTRUCTION FETCH AND IRD1
                     0022  3629 ;       3.  IF NO ERROR GO TO NEXT TEST
                     0022  3630 ;
                     0022  3631 ;
                     0022  3632 ; TEST PATTERNS:
                     0022  3633 ;
                     0022  3634 ;       NONE
                     0022  3635 ;
                     0022  3636 ;
                     0022  3637 ; LOGIC DESCRIPTION:
                     0022  3638 ;
                     0022  3639 ;       THIS TEST CHECKS THAT THE SAC GATE ARRAY IS CAPABLE OF GENERATING
                     0022  3640 ;       THE INSTRUCTION FETCH SIGNAL ON IRD1TEST.  ALSO TESTED IS THE IRD1
                     0022  3641 ;       SIGNAL ON THE VBUS WHICH IS GENERATED BY DISCRETE LOGIC.
                     0022  3642 ;
                     0022  3643 ;
                     0022  3644 ; ASSUMPTIONS:
                     0022  3645 ;
                     0022  3646 ;       THIS TEST ASSUMES THAT PREFETCH IS OPERATIONAL.
                     0022  3647 ;
                     0022  3648 ;
                     0022  3649 ; ERROR DESCRIPTION:
                     0022  3650 ;
                     0022  3651 ;       EXPECTED VBUS DATA
                     0022  3652 ;       RECEIVED VBUS DATA
                     0022  3653 ;               (BITS <7:0> = VBUS BIT POSITION)
                     0022  3654 ;               (BITS <8> = BIT STATE)
                     0022  3655 ;
                     0022  3656 ;--
                     0022  3657 ;*******************************************************************************
                     0022  3658
                     0022  3659
                     0022  3660         INITIALIZE                              ;INIT THE CPU
                     0024  3661         ERRLOOP                                 ;ERROR LOOP POINT
                     0026  3662         FETCH           0                       ;START DCS PROGRAM
                     002B  3663         BRSTCLK         3                       ;END DCS PROGRAM
                     002F  3664         CMPVBUS         1$                      ;TEST FOR INSTRUCTION FETCH
                     0034  3665         IFERROR         ,<<SAC>>,               ;NO INSTRUCTION FETCH
                     003A  3666         SKIP                                    ;GO TO NEXT TEST
                     0041  3667
```

```
                                0041  3668
                                0041  3669 BEGIN_TEST_DATA
                                0041
                                0041       ;-------------------------------------------------------------------------
                                0041  3670
                             02 0041  3671 1$:     .BYTE   ^X2
                                0042  3672         VBUSDATA        <^X18>,1                ;INSTRUCTION FETCH ON VBUS
                                0043  3673         VBUSDATA        <^X1A>,1                ;IRD1 ON VBUS
                                0044  3674
                                0044  3675
                                0044  3676 BEGIN_CPU_DATA
                                0002  3677
                                0002  3678 DCS_DATA
                                0001  3679
U 00, 7700,C800,0364,0300,0470,1801 0001  3680 ;% 00   NOP
U 01, 7700,4800,5BE4,03D8,4870,1802 000C  3684 ;% 01   PC_R[ZERO]                          ;CAUSE XB'S TO PREFETCH
U 02, 7600,4800,0364,1700,0470,1803 0017  3688 ;% 02   IRD1TEST,STROBE.V.BUS              ;CAUSE INSTRUCTION FETCH TO SET
U 03, 7300,C800,0364,0300,0470,1804 0022  3692 ;% 03   NOP,STOP.CLOCK                     ;STOP THE CPU CLOCK
                                002D  3696
                                002D  3697 CACHE_DATA
                                0001  3698
                       00000000 0001  3699         .LONG   ^X00000000                      ;VALIDATE LOCATION 0
                       00000000 0005  3700         .LONG   ^X00000000                      ;AND 4
                                0009  3701
                                0009  3702 END_CPU_DATA
                                0044  3703
                                0044  3704
                                0044  3705 END_TEST_DATA
                                0044
                                0044       ;-------------------------------------------------------------------------
                                0044  3706 ENDTEST
```

```
                        0021        .SBTTL  "TEST   073     TEST TRACE PENDING INTERRUPT
                        0021  3708 ;*******************************************************************************
                        0021  3709 ;++
                        0021  3710 ;
                        0021  3711 ; FUNCTIONAL DESCRIPTION:
                        0021  3712 ;
                        0021  3713 ;       THIS TEST CHECKS THE OPERATION OF THE CPU WHEN A T BIT TRAP OCCURS.
                        0021  3714 ;
                        0021  3715 ;
                        0021  3716 ; TEST ALGORITHM:
                        0021  3717 ;
                        0021  3718 ;       1.  EXECUTE DCS PROGRAM
                        0021  3719 ;               a.  CLEAR CONTROL P INTERRUPT FROM CON GATE ARRAY
                        0021  3720 ;               b.  SET THE TP BIT IN THE PSL
                        0021  3721 ;               c.  LOAD PC REGISTER WITH ZERO TO CAUSE PREFETCH
                        0021  3722 ;               d.  PERFORM AN IRD1TEST TO ALLOW INTERRUPT TO HAPPEN
                        0021  3723 ;       2.  TEST VBUS FOR DO SRVC AND DISABLE HI NEXT
                        0021  3724 ;       3.  IF NO ERROR TEST CSTRP REGISTER FOR CORRECT INTERRUPT VECTOR
                        0021  3725 ;       4.  IF NO ERROR GO TO NEXT TEST
                        0021  3726 ;
                        0021  3727 ;
                        0021  3728 ; TEST PATTERNS:
                        0021  3729 ;
                        0021  3730 ;       NONE
                        0021  3731 ;
                        0021  3732 ;
                        0021  3733 ; LOGIC DESCRIPTION:
                        0021  3734 ;
                        0021  3735 ;       THIS TEST VERIFIES THAT THE SAC GATE ARRAY CAN DETECT A T BIT TRAP
                        0021  3736 ;       AND GENERATE THE CORRECT INTERRUPT VECTOR.
                        0021  3737 ;
                        0021  3738 ;
                        0021  3739 ; ASSUMPTIONS:
                        0021  3740 ;
                        0021  3741 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
                        0021  3742 ;
                        0021  3743 ;
                        0021  3744 ; ERROR DESCRIPTION:
                        0021  3745 ;
                        0021  3746 ;       FIRST IFERROR:
                        0021  3747 ;               EXPECTED VBUS DATA
                        0021  3748 ;               RECEIVED VBUS DATA
                        0021  3749 ;                       (BITS <7:0> = VBUS BIT POSITION)
                        0021  3750 ;                       (BITS <8> = BIT STATE)
                        0021  3751 ;
                        0021  3752 ;       SECOND IFERROR:
                        0021  3753 ;               EXPECTED CS ADDRESS
                        0021  3754 ;               RECEIVED CS ADDRESS
                        0021  3755 ;
                        0021  3756 ;--
                        0021  3757 ;*******************************************************************************
                        0021  3758
                        0021  3759
                        0021  3760          INITIALIZE                                      ;INIT THE CPU
                        0023  3761          EXPUTRAP                                        ;EXPECT A U-TRAP
```

```
                                  0025  3762          ERRLOOP                                  ;ERROR LOOP POINT
                                  0027  3763          FETCH          0                         ;START DCS PROGRAM
                                  002C  3764          BRSTCLK        7                         ;END DCS PROGRAM
                                  0030  3765          CMPVBUS        1$                        ;TEST FOR DO SERVICE/DISABLE HI
                                  0035  3766          IFERROR        ,<<SAC><MSQ>>,            ;NO DO SERVICE/DISABLE HI NEXT
                                  003C  3767          CMPREGMSK      CSTRP,2$,,3$,,W,          ;TEST FOR TP BIT INTERRUPT
                                  0048  3768          IFERROR        ,<<SAC><MSQ>>,            ;DID NOT INTERRUPT
                                  004F  3769          SKIP                                     ;GO TO NEXT TEST
                                  0056  3770
                                  0056  3771
                                  0056  3772 BEGIN_TEST_DATA
                                  0056
                                  0056        ;-----------------------------------------------------------------------
                                  0056  3773
                            02    0056  3774 1$:      .BYTE   ^X2
                                  0057  3775          VBUSDATA       <^X19>,0                  ;DO SERVICE ON VBUS
                                  0058  3776          VBUSDATA       <^X1C>,1                  ;DISABLE HI NEXT
                          0015    0059  3777 2$:      .WORD   ^X0015                           ;TP BIT INTERRUPT
                          3FFF    005B  3778 3$:      .WORD   ^X3FFF                           ;MASK FOR CMPREGMSK PSEUDO OP
                                  005D  3779
                                  005D  3780
                                  005D  3781 BEGIN_CPU_DATA
                                  0002  3782
                                  0002  3783 DCS_DATA
                                  0001  3784
U 00, 7700,C800,0364,0300,0470,1801  0001  3785 ;% 00   NOP
U 01, 7700,4800,5BE4,0300,2470,1802  000C  3789 ;% 01   WCTRL/LOADCRAR,WB_R[TEMP0]               ;CLEAR ^P INTERRUPT
U 02, 7700,0800,5BE4,0304,2C70,1803  0017  3793 ;% 02   WCTRL/CONWRITE,WB_R[TEMP1]               ;...
U 03, 7700,4800,5BF4,0308,0070,1804  0022  3797 ;% 03   PSL_R[TEMP2]                             ;SET PSL TP BIT
U 04, 7700,C800,5BE4,03D8,4870,1805  002D  3801 ;% 04   PC_R[ZERO]                               ;CAUSE XB'S TO PREFETCH
U 05, 7700,4800,0364,1700,0470,1806  0038  3805 ;% 05   IRD1TEST                                 ;ALLOW INTERRUPT TO OCCURR
U 06, 7600,C800,0364,0300,0470,1807  0043  3809 ;% 06   NOP,STROBE.V.BUS                         ;...
U 07, 7300,C800,0364,0300,0470,1808  004E  3813 ;% 07   NOP,STOP.CLOCK                           ;STOP THE CPU CLOCK
                                  0059  3817
                                  0059  3818 RTEMP_DATA
                                  0001  3819
                        00C00000  0001  3820          .LONG   ^X00C00000                       ;ADDRESS FOR CRAR
                        00400000  0005  3821          .LONG   ^X00400000                       ;CLEAR ^P HALT
                        40000000  0009  3822          .LONG   ^X40000000                       ;SET PSL TP BIT
                                  000D  3823
                                  000D  3824 CACHE_DATA
                                  0001  3825
                        00000000  0001  3826          .LONG   ^X00000000                       ;VALIDATE LOCATION 0
                        00000000  0005  3827          .LONG   ^X00000000                       ;AND 4
                                  0009  3828
                                  0009  3829 END_CPU_DATA
                                  005D  3830
                                  005D  3831
                                  005D  3832 END_TEST_DATA
                                  005D
                                  005D        ;-----------------------------------------------------------------------
                                  005D  3833 ENDTEST
```

```
                        0020        .SBTTL  "TEST   074     TEST CONSOLE HALT INTERRUPT
                        0020  3835 ;*******************************************************************************
                        0020  3836 ;++
                        0020  3837 ;
                        0020  3838 ; FUNCTIONAL DESCRIPTION:
                        0020  3839 ;
                        0020  3840 ;       THIS TEST VERIFIES THAT THE CPU CAN DETECT AND VECTOR PROPERLY
                        0020  3841 ;       ON A CONSOLE HALT INTERRUPT.
                        0020  3842 ;
                        0020  3843 ;
                        0020  3844 ; TEST ALGORITHM:
                        0020  3845 ;
                        0020  3846 ;       1.  EXECUTE DCS PROGRAM
                        0020  3847 ;               a.  SET CONTROL P INTERRUPT IN CON GATE ARRAY
                        0020  3848 ;               b.  LOAD PC REGISTER WITH ZERO TO CAUSE PREFETCH
                        0020  3849 ;               c.  PERFORM IRD1TEST TO ALLOW INTERRUPT TO HAPPEN
                        0020  3850 ;       2.  TEST VBUS FOR CON HALT AND DO SERVICE
                        0020  3851 ;       3.  IF NO ERROR TEST CS ADDRESS FOR CORRECT INTERRUPT VECTOR
                        0020  3852 ;       4.  IF NO ERROR GO TO NEXT TEST
                        0020  3853 ;
                        0020  3854 ;
                        0020  3855 ; TEST PATTERNS:
                        0020  3856 ;
                        0020  3857 ;       NONE
                        0020  3858 ;
                        0020  3859 ;
                        0020  3860 ; LOGIC DESCRIPTION:
                        0020  3861 ;
                        0020  3862 ;       THIS TEST VERIFIES THAT THE CON GATE ARRAY CAN GENERATE A CON HALT
                        0020  3863 ;       INTERRUPT.  IT ALLOWS CHECKS THAT THE SAC GATE ARRAY CAN GENERATE
                        0020  3864 ;       THE CORRECT INTERRUPT VECTOR.
                        0020  3865 ;
                        0020  3866 ;
                        0020  3867 ; ASSUMPTIONS:
                        0020  3868 ;
                        0020  3869 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
                        0020  3870 ;
                        0020  3871 ;
                        0020  3872 ; ERROR DESCRIPTION:
                        0020  3873 ;
                        0020  3874 ;       FIRST IFERROR:
                        0020  3875 ;               EXPECTED VBUS DATA
                        0020  3876 ;               RECEIVED VBUS DATA
                        0020  3877 ;                       (BITS <7:0> = VBUS BIT POSITION)
                        0020  3878 ;                       (BITS <8> = BIT STATE)
                        0020  3879 ;
                        0020  3880 ;       SECOND IFERROR:
                        0020  3881 ;               EXPECTED CS ADDRESS
                        0020  3882 ;               RECEIVED CS ADDRESS
                        0020  3883 ;
                        0020  3884 ;--
                        0020  3885 ;*******************************************************************************
                        0020  3886
                        0020  3887
                        0020  3888         INITIALIZE                                      ;INIT THE CPU
```

```
                        0022  3889          EXPUTRAP                                ;EXPECT A U-TRAP
                        0024  3890          ERRLOOP                                 ;ERROR LOOP POINT
                        0026  3891          FETCH           0                       ;START DCS PROGRAM
                        002B  3892          BRSTCLK         6                       ;END DCS PROGRAM
                        002F  3893          CMPVBUS         1$                      ;TEST FOR DO SERVICE/CON HALT
                        0034  3894          IFERROR         ,<<CON><SAC>>,<UBI>     ;NO DO SERVICE/CON HALT
                        003C  3895          CMPREGMSK       CSTRP,2$,,3$,,W,        ;TEST FOR CONSOLE HALT INT
                        0048  3896          IFERROR         ,<<SAC>>,               ;DID NOT INTERRUPT
                        004E  3897          SKIP                                    ;GO TO NEXT TEST
                        0055  3898
                        0055  3899
                        0055  3900 BEGIN_TEST_DATA
                        0055
                        0055              ;-------------------------------------------------------------------------
                        0055  3901
                     02 0055  3902 1$:      .BYTE   ^X2
                        0056  3903          VBUSDATA        <^X07>,0                ;CON HALT ON VBUS
                        0057  3904          VBUSDATA        <^X19>,0                ;DO SERVICE ON VBUS
                   0016 0058  3905 2$:      .WORD   ^X0016                          ;CONSOLE HALT INTERRUPT
                   3FFF 005A  3906 3$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                        005C  3907
                        005C  3908
                        005C  3909 BEGIN_CPU_DATA
                        0002  3910
                        0002  3911 DCS_DATA
                        0001  3912
U 00, 7700,C800,0364,0300,0470,1801     0001  3913 ;% 00   NOP
U 01, 7700,4800,5BE4,0300,2470,1802     000C  3917 ;% 01   WCTRL/LOADCRAR,WB_R[TEMP0]          ;SET ^P INTERRUPT
U 02, 7700,0800,5BE4,0304,2C70,1803     0017  3921 ;% 02   WCTRL/CONWRITE,WB_R[TEMP1]         ;...
U 03, 7700,4800,5BE4,03D8,4870,1804     0022  3925 ;% 03   PC_R[ZERO]                         ;CAUSE XB'S TO PREFETCH
U 04, 7700,4800,0364,1700,0470,1805     002D  3929 ;% 04   IRD1TEST                          ;ALLOW INTERRUPT TO OCCUR
U 05, 7600,4800,0364,0300,0470,1806     0038  3933 ;% 05   NOP,STROBE.V.BUS                  ;...
U 06, 7300,C800,0364,0300,0470,1807     0043  3937 ;% 06   NOP,STOP.CLOCK                    ;STOP THE CPU CLOCK
                        004E  3941
                        004E  3942 RTEMP_DATA
                        0001  3943
                00C00000 0001  3944          .LONG   ^X00C00000                      ;ADDRESS FOR CRAR
                00900000 0005  3945          .LONG   ^X00900000                      ;SET ^P HALT
                        0009  3946
                        0009  3947 CACHE_DATA
                        0001  3948
                00000000 0001  3949          .LONG   ^X00000000                      ;VALIDATE LOCATION 0
                00000000 0005  3950          .LONG   ^X00000000                      ;AND 4
                        0009  3951
                        0009  3952 END_CPU_DATA
                        005C  3953
                        005C  3954
                        005C  3955 END_TEST_DATA
                        005C
                        005C              ;-------------------------------------------------------------------------
                        005C  3956 ENDTEST
```

```
                        0020          .SBTTL  "TEST   075     TEST NEXT INTERVAL REGISTER
                        0020  3958 ;*******************************************************************************
                        0020  3959 ;++
                        0020  3960 ;
                        0020  3961 ; FUNCTIONAL DESCRIPTION:
                        0020  3962 ;
                        0020  3963 ;       THIS TEST CHECKS THE DATA INTEGRITY OF THE INTERNAL NEXT INTERVAL
                        0020  3964 ;       REGISTER.
                        0020  3965 ;
                        0020  3966 ;
                        0020  3967 ; TEST ALGORITHM:
                        0020  3968 ;
                        0020  3969 ;       1.  CREATE A TEST LOOP OF 5
                        0020  3970 ;       2.  LOAD A TEST PATTERN INTO THE RDM WDREG
                        0020  3971 ;       3.  EXECUTE DCS PROGRAM
                        0020  3972 ;               a.  WRITE TEST PATTERN TO INIR REGISTER
                        0020  3973 ;               b.  READ TEST PATTERN BACK TO RDM
                        0020  3974 ;       4.  TEST INIR REGISTER FOR CORRECT TEST PATTERN
                        0020  3975 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
                        0020  3976 ;
                        0020  3977 ;
                        0020  3978 ; TEST PATTERNS:
                        0020  3979 ;
                        0020  3980 ;       ITERATION       TEST PATTERN
                        0020  3981 ;           1               AAAA
                        0020  3982 ;           2               5555
                        0020  3983 ;           3               3333
                        0020  3984 ;           4               0F0F
                        0020  3985 ;           5               00FF
                        0020  3986 ;
                        0020  3987 ;
                        0020  3988 ; LOGIC DESCRIPTION:
                        0020  3989 ;
                        0020  3990 ;       THIS TEST VERIFIES THE DATA INTEGRITY OF THE INTERNAL NEXT INTERVAL
                        0020  3991 ;       REGISTER LOCATED IN THE TOK GATE ARRAY.
                        0020  3992 ;
                        0020  3993 ;
                        0020  3994 ; ASSUMPTIONS:
                        0020  3995 ;
                        0020  3996 ;       THIS TEST ASSUMES THE WBUS IS OPERATIONAL.
                        0020  3997 ;
                        0020  3998 ;
                        0020  3999 ; ERROR DESCRIPTION:
                        0020  4000 ;
                        0020  4001 ;       EXPECTED INIR CONTENTS
                        0020  4002 ;       RECEIVED INIR CONTENTS
                        0020  4003 ;
                        0020  4004 ;--
                        0020  4005 ;*******************************************************************************
                        0020  4006
                        0020  4007
                        0020  4008          INITIALIZE                              ;INIT THE CPU
                        0022  4009          LOOP            I,1,5                   ;TEST LOOP OF 5
                        0029  4010          LOADREG         WDREG,1$,I,L            ;TEST PATTERN TO WD REGISTER
                        0031  4011          ERRLOOP                                 ;ERROR LOOP POINT
```

```
                                  0033  4012          FETCH          0                        ;START DCS PROGRAM
                                  0038  4013          BRSTCLK        3                        ;END DCS PROGRAM
                                  003C  4014          CMPREGMSK      WHREG,1$,I,2$,,L,        ;TEST FOR CORRECT TEST PATTERN
                                  0048  4015          IFERROR        ,<<TOK>>,                ;INCORRECT TEST PATTERN
                                  004E  4016          ENDLOOP        I                        ;END TEST LOOP
                                  0051  4017          SKIP                                    ;GO TO NEXT TEST
                                  0058  4018
                                  0058  4019
                                  0058  4020 BEGIN_TEST_DATA
                                  0058
                                  0058       ;-----------------------------------------------------------------------
                                  0058  4021
                        0000AAAA  0058  4022 1$:      .LONG   ^X0000AAAA                      ;TEST PATTERN TABLE
                        00005555  005C  4023          .LONG   ^X00005555
                        00003333  0060  4024          .LONG   ^X00003333
                        00000F0F  0064  4025          .LONG   ^X00000F0F
                        000000FF  0068  4026          .LONG   ^X000000FF
                        0000FFFF  006C  4027 2$:      .LONG   ^X0000FFFF                      ;MASK FOR CMPREGMSK PSEUDO OP
                                  0070  4028
                                  0070  4029
                                  0070  4030 BEGIN_CPU_DATA
                                  0002  4031
                                  0002  4032 DCS_DATA
                                  0001  4033
U 00, 7700,C800,0364,0300,0470,1801  0001  4034 ;% 00    NOP
U 01, 5700,4800,0364,0300,1470,1802  000C  4038 ;% 01    WCTRL/INIR_WB,WB_RDM                    ;TEST PATTERN TO INIR REGISTER
U 02, 6700,4800,0364,0300,1C70,1803  0017  4042 ;% 02    RDM_WB,WCTRL/INIR                       ;READ INIR TO RDM
U 03, 7300,C800,0364,0300,0470,1804  0022  4046 ;% 03    NOP,STOP.CLOCK                          ;STOP THE CPU CLOCK
                                  002D  4050
                                  002D  4051 END_CPU_DATA
                                  0070  4052
                                  0070  4053
                                  0070  4054 END_TEST_DATA
                                  0070
                                  0070       ;-----------------------------------------------------------------------
                                  0070  4055 ENDTEST
```

```
                        0021        .SBTTL  "TEST   076     TEST INTERVAL COUNT REGISTER
                        0021  4057 ;*****************************************************************************
                        0021  4058 ;++
                        0021  4059 ;
                        0021  4060 ; FUNCTIONAL DESCRIPTION:
                        0021  4061 ;
                        0021  4062 ;       THIS TEST CHECKS THE DATA INTEGRITY OF THE INTERNAL INTERVAL COUNT
                        0021  4063 ;       REGISTER.
                        0021  4064 ;
                        0021  4065 ;
                        0021  4066 ; TEST ALGORITHM:
                        0021  4067 ;
                        0021  4068 ;       1.  CREATE A TEST LOOP OF 5
                        0021  4069 ;       2.  LOAD A TEST PATTERN INTO THE RDM WDREG
                        0021  4070 ;       3.  EXECUTE DCS PROGRAM
                        0021  4071 ;               a.  WRITE TEST PATTERN INTO INIR REGISTER
                        0021  4072 ;               b.  TEST T BIT IN TCSR REGISTER
                        0021  4073 ;               c.  READ TCSR.IICR REGISTERS BACK TO THE RDM
                        0021  4074 ;       4.  TEST TCSR.IICR REGISTERS FOR CORRECT DATA
                        0021  4075 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
                        0021  4076 ;
                        0021  4077 ;
                        0021  4078 ; TEST PATTERNS:
                        0021  4079 ;
                        0021  4080 ;       NONE
                        0021  4081 ;
                        0021  4082 ;
                        0021  4083 ; LOGIC DESCRIPTION:
                        0021  4084 ;
                        0021  4085 ;       THIS TEST CHECKS THE DATA INTEGRITY OF THE INTERNAL INTERVAL COUNTER
                        0021  4086 ;       REGISTER IN THE TOK GATE ARRAY.  SINCE THIS REGISTER CANNOT BE
                        0021  4087 ;       DIRECTLY LOADED YOU MUST FIRST LOAD THE INIR THEN SET THE TRANSFER
                        0021  4088 ;       BIT (T) IN THE TIMER CONTROL AND STATUS REGISTER (TCSR).  WHEN YOU
                        0021  4089 ;       READ THE IICR BACK YOU GET THE TCSR IN THE UPPER WORD.  ANY FAILURES
                        0021  4090 ;       SHOULD BE CONFINED TO TOK.
                        0021  4091 ;
                        0021  4092 ;
                        0021  4093 ; ASSUMPTIONS:
                        0021  4094 ;
                        0021  4095 ;       THIS TEST ASSUMES THE WBUS IS OPERATIONAL.
                        0021  4096 ;
                        0021  4097 ;
                        0021  4098 ; ERROR DESCRIPTION:
                        0021  4099 ;
                        0021  4100 ;       EXPECTED TCSR.IICR CONTENTS
                        0021  4101 ;       RECEIVED TCSR.IICR CONTENTS
                        0021  4102 ;
                        0021  4103 ;--
                        0021  4104 ;*****************************************************************************
                        0021  4105
                        0021  4106
                        0021  4107         INITIALIZE                              ;INIT THE CPU
                        0023  4108         LOOP            I,1,5                   ;TEST LOOP OF 5
                        002A  4109         LOADREG         WDREG,1$,I,L            ;LOAD TEST PATTERN INTO WD REG
                        0032  4110         ERRLOOP                                 ;ERROR LOOP POINT
```

```
                         0034  4111          FETCH           0                       ;START DCS PROGRAM
                         0039  4112          BRSTCLK         4                       ;END DCS PROGRAM
                         003D  4113          CMPREGMSK       WHREG,1$,I,2$,,L,       ;TEST FOR CORRECT PATTERN
                         0049  4114          IFERROR         ,<TOK>,                 ;INCORRECT PATTERN
                         004F  4115          ENDLOOP         I                       ;END TEST LOOP
                         0052  4116          SKIP                                    ;GO TO NEXT TEST
                         0059  4117
                         0059  4118
                         0059  4119 BEGIN_TEST_DATA
                         0059
                         0059          ;-------------------------------------------------------------------------
                         0059  4120
                0014AAAA 0059  4121 1$:      .LONG    ^X0014AAAA                     ;TEST PATTERN TABLE
                00145555 005D  4122          .LONG    ^X00145555
                00143333 0061  4123          .LONG    ^X00143333
                00140F0F 0065  4124          .LONG    ^X00140F0F
                001400FF 0069  4125          .LONG    ^X001400FF
                80FFFFFF 006D  4126 2$:      .LONG    ^X80FFFFFF                     ;MASK FOR CMPREGMSK PSEUDO OP
                         0071  4127
                         0071  4128
                         0071  4129 BEGIN_CPU_DATA
                         0002  4130
                         0002  4131 DCS_DATA
                         0001  4132
U 00, 7700,C800,0364,0300,0470,1801  0001  4133 ;% 00    NOP
U 01, 5700,4800,0364,0300,1470,1802  000C  4137 ;% 01    WCTRL/INIR_WB,WB_RDM           ;TEST PATTERN TO INIR REGISTER
U 02, 7700,4800,5BE4,0300,1670,1803  0017  4141 ;% 02    WCTRL/TCSR_WB,WB_R[TEMP0]      ;LOAD T BIT INTO TCSR REGISTER
U 03, 6700,4800,0364,0300,1E70,1804  0022  4145 ;% 03    RDM_WB,WCTRL/TCSR.IICR         ;READ TEST PATTERN BACK TO RDM
U 04, 7300,4800,0364,0300,0470,1805  002D  4149 ;% 04    NOP,STOP.CLOCK                 ;STOP THE CPU CLOCK
                         0038  4153
                         0038  4154 RTEMP_DATA
                         0001  4155
                00100000 0001  4156          .LONG    ^X00100000                     ;T BIT FOR TCSR REGISTER
                         0005  4157
                         0005  4158 END_CPU_DATA
                         0071  4159
                         0071  4160
                         0071  4161 END_TEST_DATA
                         0071
                         0071          ;-------------------------------------------------------------------------
                         0071  4162 ENDTEST
```

```
0023        .SBTTL  "TEST   077     TIMER SERVICE INTERRUPT TEST 1
0023  4164 ;*******************************************************************************
0023  4165 ;++
0023  4166 ;
0023  4167 ; FUNCTIONAL DESCRIPTION:
0023  4168 ;
0023  4169 ;       THIS TEST VERIFIES THAT THE TOK GATE ARRAY CAN GENERATE A TIMER SERVICE
0023  4170 ;       INTERRUPT WHEN THE TR BIT IS SET IN THE TCSR REGISTER.
0023  4171 ;
0023  4172 ;
0023  4173 ; TEST ALGORITHM:
0023  4174 ;
0023  4175 ;       SUBTEST #1
0023  4176 ;       1.  LOAD MICRO CODE FOR SUBTEST #1
0023  4177 ;       2.  EXECUTE DCS PROGRAM
0023  4178 ;               a.  SET THE T BIT IN THE TCSR REGISTER
0023  4179 ;               b.  LOAD PC REGISTER WITH ZERO TO CAUSE PREFETCH
0023  4180 ;               c.  PERFORM IRD1TEST TO ALLOW INTERRUPT TO HAPPEN
0023  4181 ;       3.  TEST VBUS FOR DO SRVC
0023  4182 ;       4.  IF NO ERROR TEST CSTRP REGISTER FOR CORRECT INTERRUPT VECTOR
0023  4183 ;       5.  IF NO ERROR GO TO SUBTEST 2
0023  4184 ;
0023  4185 ;       SUBTEST #2
0023  4186 ;       1.  LOAD MICRO CODE FOR SUBTEST #2
0023  4187 ;       2.  EXECUTE DCS PROGRAM
0023  4188 ;               a.  SET THE T BIT IN THE TCSR REGISTER
0023  4189 ;               b.  PERFORM A NOP TO ALLOW INTERRUPT TIME TO SET UP
0023  4190 ;               c.  BRANCH ON TIMER SERVICE INTERRUPT
0023  4191 ;       3.  TEST CS ADDRESS FOR CORRECT BRANCH
0023  4192 ;       4.  IF NO ERROR GO TO NEXT TEST
0023  4193 ;
0023  4194 ;
0023  4195 ; TEST PATTERNS:
0023  4196 ;
0023  4197 ;       NONE
0023  4198 ;
0023  4199 ;
0023  4200 ; LOGIC DESCRIPTION:
0023  4201 ;
0023  4202 ;       THIS TEST SETS THE TRANSFER BIT (T) IN THE TCSR REGISTER WHICH IN TURN
0023  4203 ;       SET THE TRANSFER REQUEST BIT (TR).  SETTING TR SHOULD CAUSE THE TOK
0023  4204 ;       GATE ARRAY TO GENERATE A TIMER SERVICE INTERRUPT.  THE SAC GATE ARRAY
0023  4205 ;       IS RESPONSIBLE FOR HANDLING THE INTERRUPT.
0023  4206 ;       SUBTEST 2 CHECKS THE BRANCHING LOGIC ON THE DPM MODULE THAT GATES THE
0023  4207 ;       TIMER SERVICE SIGNAL INTO CS ADDRESS BIT 0.
0023  4208 ;
0023  4209 ;
0023  4210 ; ASSUMPTIONS:
0023  4211 ;
0023  4212 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
0023  4213 ;
0023  4214 ;
0023  4215 ; ERROR DESCRIPTION:
0023  4216 ;
0023  4217 ;       SUBTEST #1:
```

```
                     0023  4218 ;       FIRST IFERROR:
                     0023  4219 ;               EXPECTED VBUS DATA
                     0023  4220 ;               RECEIVED VBUS DATA
                     0023  4221 ;                       (BITS <7:0> = VBUS BIT POSITION)
                     0023  4222 ;                       (BITS <8> = BIT STATE)
                     0023  4223 ;
                     0023  4224 ;       SECOND IFERROR:
                     0023  4225 ;               EXPECTED CS ADDRESS
                     0023  4226 ;               RECEIVED CS ADDRESS
                     0023  4227 ;
                     0023  4228 ;       SUBTEST #2:
                     0023  4229 ;               EXPECTED CS ADDRESS
                     0023  4230 ;               RECEIVED CS ADDRESS
                     0023  4231 ;
                     0023  4232 ;--
                     0023  4233 ;******************************************************************************
                     0023  4234 ;//////////////////////////////////////////////////////////////////////////////
                     0023  4235 ;+
                     0023  4236 ;SUBTEST #1
                     0023  4237 ;
                     0023  4238 ;       TEST FOR TIMER SERVICE INTERRUPT
                     0023  4239 ;-
                     0023  4240         SUBTEST                                 ;SUBTEST #1
                     0026
                     0026       ;//////////////////////////////////////////////////////////////////////////////
                     0026  4241         INITIALIZE                              ;INIT THE CPU
                     0028  4242         EXPUTRAP                                ;EXPECT A U-TRAP
                     002A  4243         LOADDCS         4$,,02,4                ;LOAD MICRO CODE FOR SUBTEST
                     0031  4244         ERRLOOP                                 ;ERROR LOOP POINT
                     0033  4245         FETCH           0                       ;START DCS PROGRAM
                     0038  4246         BRSTCLK         5                       ;END DCS PROGRAM
                     003C  4247         CMPVBUS         3$                      ;TEST FOR DO SRVC
                     0041  4248         IFERROR         ,<<TOK><SAC>>,          ;NOT ON VBUS
                     0048  4249         CMPREGMSK       CSTRP,1$,,2$,,W,        ;TEST FOR CORRECT INT ADDRESS
                     0054  4250         IFERROR         ,<<SAC><MSQ>>,          ;NOT CORRECT INTERRUPT
                     005B  4251
                     005B  4252
                     005B  4253 ;//////////////////////////////////////////////////////////////////////////////
                     005B  4254 ;+
                     005B  4255 ;SUBTEST #2
                     005B  4256 ;
                     005B  4257 ;       TEST BRANCH ON TIMER SERVICE
                     005B  4258 ;-
                     005B  4259         SUBTEST                                 ;SUBTEST #2
                     005E
                     005E       ;//////////////////////////////////////////////////////////////////////////////
                     005E  4260         INITIALIZE                              ;INIT THE CPU
                     0060  4261         LOADDCS         5$,,02,3                ;LOAD MICRO CODE FOR SUBTEST
                     0067  4262         ERRLOOP                                 ;ERROR LOOP POINT
                     0069  4263         FETCH           0                       ;START DCS PROGRAM
                     006E  4264         BRSTCLK         4                       ;END DCS PROGRAM
                     0072  4265         CMPREGMSK       CSTRP,6$,,2$,,W,        ;TEST CS ADDRESS
                     007E  4266         IFERROR                                 ;INCORRECT BRANCH
                     0083  4267         SKIP                                    ;GO TO NEXT TEST
                     008A  4268
```

```
                                    008A  4269
                                    008A  4270 BEGIN_TEST_DATA
                                    008A
                                    008A       ;-------------------------------------------------------------------------
                                    008A  4271
                              0014  008A  4272 1$:     .WORD   ^X0014                          ;ADDRESS OF TIMER SERVICE INT
                              3FFF  008C  4273 2$:     .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                                01  008E  4274 3$:     .BYTE   ^X1
                                    008F  4275          VBUSDATA        <^X19>,0            ;DO SRVC ON VBUS
                                    0090  4276 4$:                                          ;MICRO CODE SUBTEST #1
U 02, 7700,C800,5BE4,03D8,4870,1803 0090  4277 ;% 02   PC_R[ZERO]                           ;CAUSE XB'S TO PREFETCH
U 03, 7700,C800,0364,1700,0470,1804 009B  4281 ;% 03   IRD1TEST                             ;ALLOW INTERRUPT TO HAPPEN
U 04, 7600,4800,0364,0300,0470,1805 00A6  4285 ;% 04   NOP,STROBE.V.BUS                      ;CATCH DO SRVC ON VBUS
U 05, 7300,4800,0364,0300,0470,1806 00B1  4289 ;% 05   NOP,STOP.CLOCK                       ;STOP THE CPU CLOCK
                                    00BC  4293 5$:                                          ;MICRO CODE SUBTEST #2
U 02, 7700,4800,0364,0300,0470,1803 00BC  4294 ;% 02   NOP
U 03, 7700,4800,0364,AF00,0470,1800 00C7  4298 ;% 03   BUT/INT-TIMSERV,NEXT/1800             ;BRANCH ON TIMER SERVICE
U 04, 7300,4800,0364,0300,0470,1805 00D2  4302 ;% 04   NOP,STOP.CLOCK                       ;STOP THE CPU CLOCK
                              1803  00DD  4306 6$:     .WORD   ^X1803                      ;BRANCH ADDRESS
                                    00DF  4307
                                    00DF  4308 BEGIN_CPU_DATA
                                    0002  4309
                                    0002  4310 DCS_DATA
                                    0001  4311
U 00, 7700,C800,0364,0300,0470,1801 0001  4312 ;% 00   NOP
U 01, 7700,C800,5BE4,0300,1670,1802 000C  4316 ;% 01   WCTRL/TCSR_WB,WB_R[TEMP0]             ;SET T BIT IN TCSR REGISTER
U 02, 7700,C800,5BE4,03D8,4870,1803 0017  4320 ;% 02   PC_R[ZERO]                           ;CAUSE XB'S TO PREFETCH
U 03, 7700,C800,0364,1700,0470,1804 0022  4324 ;% 03   IRD1TEST                             ;ALLOW INTERRUPT TO HAPPEN
U 04, 7600,4800,0364,0300,0470,1805 002D  4328 ;% 04   NOP,STROBE.V.BUS                      ;CATCH DO SRVC ON VBUS
U 05, 7300,4800,0364,0300,0470,1806 0038  4332 ;% 05   NOP,STOP.CLOCK                       ;STOP THE CPU CLOCK
                                    0043  4336
                                    0043  4337 RTEMP_DATA
                                    0001  4338
                         00100000   0001  4339          .LONG   ^X00100000                ;SET T BIT IN TCSR REGISTER
                                    0005  4340
                                    0005  4341 CACHE_DATA
                                    0001  4342
                         00000000   0001  4343          .LONG   ^X00000000                ;VALIDATE CACHE LOCATIONS 0
                         00000000   0005  4344          .LONG   ^X00000000                ;AND 4
                                    0009  4345
                                    0009  4346 END_CPU_DATA
                                    00DF  4347
                                    00DF  4348 END_TEST_DATA
                                    00DF
                                    00DF       ;-------------------------------------------------------------------------
                                    00DF  4349 ENDTEST
```

```
                        0023       .SBTTL  "TEST   078     TIMER SERVICE INTERRUPT TEST 2
                        0023  4351 ;*******************************************************************************
                        0023  4352 ;++
                        0023  4353 ;
                        0023  4354 ; FUNCTIONAL DESCRIPTION:
                        0023  4355 ;
                        0023  4356 ;       THIS TEST VERFIES THAT A TIMER SERVICE INTERRUPT IS GENERATED WHEN
                        0023  4357 ;       THE SR BIT IS SET IN THE TCSR REGISTER.
                        0023  4358 ;
                        0023  4359 ;
                        0023  4360 ; TEST ALGORITHM:
                        0023  4361 ;
                        0023  4362 ;       1.  EXECUTE DCS PROGRAM
                        0023  4363 ;               a.  LOAD THE INIR REGISTER WITH ONES
                        0023  4364 ;               b.  SET T BIT IN TCSR REGISTER TO TRANSFER INIR TO IICR
                        0023  4365 ;               c.  STEP IICR REGISTER BY ONE
                        0023  4366 ;               d.  READ THE TCSR.IICR REGISTERS TO THE RDM
                        0023  4367 ;               e.  LOAD THE PC REGISTER WITH ZERO TO CAUSE PREFETCH
                        0023  4368 ;               f.  PERFORM IRD1TEST TO ALLOW INTERRUPT TO HAPPEN
                        0023  4369 ;       2.  TEST TCSR.IICR REGISTERS FOR CORRECT DATA
                        0023  4370 ;       3.  IF NO ERROR TEST VBUS FOR DO SRVC
                        0023  4371 ;       4.  IF NO ERROR TEST CS ADDRESS
                        0023  4372 ;       5.  IF NO ERROR GO TO NEXT TEST
                        0023  4373 ;
                        0023  4374 ;
                        0023  4375 ; TEST PATTERNS:
                        0023  4376 ;
                        0023  4377 ;       NONE
                        0023  4378 ;
                        0023  4379 ;
                        0023  4380 ; LOGIC DESCRIPTION:
                        0023  4381 ;
                        0023  4382 ;       THIS TEST CHECKS THAT SETTING THE SERVICE REQUEST BIT (SR) IN THE
                        0023  4383 ;       TCSR REGISTER WILL CAUSE A TIMER SERVICE INTERRUPT.  THE INTERRUPT
                        0023  4384 ;       SHOULD BE HANDLED BY THE SAC GATE ARRAY.
                        0023  4385 ;
                        0023  4386 ;
                        0023  4387 ; ASSUMPTIONS:
                        0023  4388 ;
                        0023  4389 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
                        0023  4390 ;
                        0023  4391 ;
                        0023  4392 ; ERROR DESCRIPTION:
                        0023  4393 ;
                        0023  4394 ;       FIRST IFERROR:
                        0023  4395 ;               EXPECTED TCSR.IICR CONTENTS
                        0023  4396 ;               RECEIVED TCSR.IICR CONTENTS
                        0023  4397 ;
                        0023  4398 ;       SECOND IFERROR:
                        0023  4399 ;               EXPECTED VBUS DATA
                        0023  4400 ;               RECEIVED VBUS DATA
                        0023  4401 ;                       (BITS <7:0> = VBUS BIT POSITION)
                        0023  4402 ;                       (BITS <8> = BIT STATE)
                        0023  4403 ;
                        0023  4404 ;       THIRD IFERROR:
```

```
                        0023  4405 ;                EXPECTED CS ADDRESS
                        0023  4406 ;                RECEIVED CS ADDRESS
                        0023  4407 ;
                        0023  4408 ;--
                        0023  4409 ;*****************************************************************************
                        0023  4410
                        0023  4411
                        0023  4412          INITIALIZE                              ;INIT THE CPU
                        0025  4413          EXPUTRAP                                ;EXPECT A U-TRAP
                        0027  4414          ERRLOOP                                 ;ERROR LOOP POINT
                        0029  4415          FETCH           0                       ;START DCS PROGRAM
                        002E  4416          BRSTCLK         8                       ;END DCS PROGRAM
                        0032  4417          CMPREGMSK       WHREG,4$,,5$,,L,        ;TEST TCSR.IICR CONTENTS
                        003E  4418          IFERROR         ,<TOK>,                 ;REGISTERS NOT SETUP RIGHT
                        0044  4419          CMPVBUS         3$                      ;TEST FOR DO SRVC
                        0049  4420          IFERROR         ,<<TOK><SAC>>,          ;NO DO SRVC ON VBUS
                        0050  4421          CMPREGMSK       CSTRP,1$,,2$,,W,        ;TEST FOR INTERRUPT ADDRESS
                        005C  4422          IFERROR         ,<<SAC><MSQ>>,          ;INCORRECT ADDRESS
                        0063  4423          SKIP                                    ;GO TO NEXT TEST
                        006A  4424
                        006A  4425
                        006A  4426 BEGIN_TEST_DATA
                        006A              ;-------------------------------------------------------------------
                        006A  4427
                  0014  006A  4428 1$:      .WORD   ^X0014                          ;TIMER SERVICE INT ADDRESS
                  3FFF  006C  4429 2$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                    01  006E  4430 3$:      .BYTE   ^X1
                        006F  4431          VBUSDATA        <^X19>,0                ;DO SRVC ON VBUS
              00080000  0070  4432 4$:      .LONG   ^X00080000                      ;TCSR.IICR CONTENTS
              80FFFFFF  0074  4433 5$:      .LONG   ^X80FFFFFF                      ;MASK FOR CMPREGMSK PSEUDO OP
                        0078  4434
                        0078  4435
                        0078  4436 BEGIN_CPU_DATA
                        0002  4437
                        0002  4438 DCS_DATA
                        0001  4439
U 00, 7700,C800,0364,0300,0470,1801     0001  4440 ;% 00    NOP
U 01, 7700,4800,5BE4,0300,1470,1802     000C  4444 ;% 01    WCTRL/INIR_WB,WB_R[TEMP0]       ;LOAD INIR REGISTER WITH ONE'S
U 02, 7700,0800,5BE4,0304,1670,1803     0017  4448 ;% 02    WCTRL/TCSR_WB,WB_R[TEMP1]       ;SET UP TCSR REGISTER
U 03, 7700,8800,5BE4,0308,1670,1804     0022  4452 ;% 03    WCTRL/TCSR_WB,WB_R[TEMP2]       ;INCREMENT IICR REGISTER
U 04, 6700,C800,0364,0300,1E70,1805     002D  4456 ;% 04    RDM_WB,WCTRL/TCSR.IICR          ;READ TCSR.IICR TO RDM
U 05, 7700,C800,5BE4,03D8,4870,1806     0038  4460 ;% 05    PC_R[ZERO]                      ;CAUSE PC TO PREFETCH
U 06, 7700,C800,0364,1700,0470,1807     0043  4464 ;% 06    IRD1TEST                        ;ALLOW INT TO HAPPEN
U 07, 7600,C800,0364,0300,0470,1808     004E  4468 ;% 07    NOP,STROBE.V.BUS                ;STROBE THE VBUS
U 08, 7300,4800,0364,0300,0470,1809     0059  4472 ;% 08    NOP,STOP.CLOCK                  ;STOP THE CPU CLOCK
                        0064  4476
                        0064  4477 RTEMP_DATA
                        0001  4478
              0000FFFF  0001  4479          .LONG   ^X0000FFFF                      ;DATA FOR INIR REGISTER
              00100000  0005  4480          .LONG   ^X00100000                      ;DATA FOR TCSR REGISTER
              00240000  0009  4481          .LONG   ^X00240000                      ;STEP THE TIMER
                        000D  4482
                        000D  4483 CACHE_DATA
                        0001  4484
```

```
                    00000000  0001  4485          .LONG   ^X00000000                      ;VALIDATE LOCATION 0
                    00000000  0005  4486          .LONG   ^X00000000                      ;AND 4
                              0009  4487
                              0009  4488 END_CPU_DATA
                              0078  4489
                              0078  4490
                              0078  4491 END_TEST_DATA
                              0078
                              0078        ;-----------------------------------------------------------------------
                              0078  4492 ENDTEST
```

```
0023          .SBTTL  "TEST   079     TIMER SERVICE INTERRUPT TEST 3
0023  4494 ;*****************************************************************************
0023  4495 ;++
0023  4496 ;
0023  4497 ; FUNCTIONAL DESCRIPTION:
0023  4498 ;
0023  4499 ;       THIS TEST CHECKS THAT A TIMER SERVICE INTERRUPT IS GENERATED WHEN
0023  4500 ;       TR AND SR ARE SET IN THE TCSR REGISTER.
0023  4501 ;
0023  4502 ;
0023  4503 ; TEST ALGORITHM:
0023  4504 ;
0023  4505 ;       1.  EXECUTE DCS PROGRAM
0023  4506 ;               a.  LOAD INIR REGISTER WITH ONES
0023  4507 ;               b.  SET T BIT IN TCSR REGISTER TO TRANSFER INIR TO IICR
0023  4508 ;               c.  STEP THE IICR REGISTER BY ONE
0023  4509 ;               d.  READ THE TCSR.IICR REGISTERS BACK TO THE RDM
0023  4510 ;               e.  LOAD THE PC WITH ZERO TO CAUSE THE XBS TO PREFETCH
0023  4511 ;               f.  PERFORM AND IRD1TEST TO ALLOW THE INTERRUPT TO HAPPEN
0023  4512 ;       2.  TEST THE TCSR.IICR REGISTER FOR CORRECT DATA
0023  4513 ;       3.  IF NO ERROR TEST THE VBUS FOR DO SRVC
0023  4514 ;       4.  IF NO ERROR TEST THE CS ADDRESS FOR THE CORRECT INTERRUPT VECTOR
0023  4515 ;       5.  IF NO ERROR GO TO THE NEXT TEST
0023  4516 ;
0023  4517 ;
0023  4518 ; TEST PATTERNS:
0023  4519 ;
0023  4520 ;       NONE
0023  4521 ;
0023  4522 ;
0023  4523 ; LOGIC DESCRIPTION:
0023  4524 ;
0023  4525 ;       THIS TEST INSURES THAT THE TOK GATE ARRAY CAN GENERATE A TIMER SERVICE
0023  4526 ;       INTERRUPT WHEN THE SERVICE REQUEST (SR) AND TRANSFER REQUEST (TR) BITS
0023  4527 ;       ARE SET IN THE TCSR REGISTER.  THE SAC GATE ARRAY IS RESPONSIBLE FOR
0023  4528 ;       HANDLING THE INTERRUPT AND GENERATING THE PROPER INTERRUPT VECTOR.
0023  4529 ;
0023  4530 ;
0023  4531 ; ASSUMPTIONS:
0023  4532 ;
0023  4533 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
0023  4534 ;
0023  4535 ;
0023  4536 ; ERROR DESCRIPTION:
0023  4537 ;
0023  4538 ;       FIRST IFERROR:
0023  4539 ;               EXPECTED TCSR.IICR CONTENTS
0023  4540 ;               RECEIVED TCSR.IICR CONTENTS
0023  4541 ;
0023  4542 ;       SECOND IFERROR:
0023  4543 ;               EXPECTED VBUS DATA
0023  4544 ;               RECEIVED VBUS DATA
0023  4545 ;                       (BITS <7:0> = VBUS BIT POSITION)
0023  4546 ;                       (BITS <8> = BIT STATE)
0023  4547 ;
```

```
                        0023  4548 ;       THIRD IFERROR:
                        0023  4549 ;               EXPECTED CS ADDRESS
                        0023  4550 ;               RECEIVED CS ADDRESS
                        0023  4551 ;
                        0023  4552 ;--
                        0023  4553 ;*****************************************************************************
                        0023  4554
                        0023  4555
                        0023  4556         INITIALIZE                              ;INIT THE CPU
                        0025  4557         EXPUTRAP                                ;EXPECT A U-TRAP
                        0027  4558         ERRLOOP                                 ;ERROR LOOP POINT
                        0029  4559         FETCH           0                       ;START DCS PROGRAM
                        002E  4560         BRSTCLK         8                       ;END DCS PROGRAM
                        0032  4561         CMPREGMSK       WHREG,4$,,5$,,L,        ;TEST TCSR.IICR CONTENTS
                        003E  4562         IFERROR         ,<TOK>,                 ;REGISTERS NOT SETUP RIGHT
                        0044  4563         CMPVBUS         3$                      ;TEST FOR DO SRVC
                        0049  4564         IFERROR         ,<<TOK><SAC>>,          ;NO DO SRVC ON VBUS
                        0050  4565         CMPREGMSK       CSTRP,1$,,2$,,W,        ;TEST FOR INTERRUPT ADDRESS
                        005C  4566         IFERROR         ,<<SAC><MSQ>>,          ;INCORRECT ADDRESS
                        0063  4567         SKIP                                    ;GO TO NEXT TEST
                        006A  4568
                        006A  4569
                        006A  4570 BEGIN_TEST_DATA
                        006A              ;-------------------------------------------------------------------
                        006A  4571
                0014    006A  4572 1$:      .WORD   ^X0014                        ;TIMER SERVICE INT ADDRESS
                3FFF    006C  4573 2$:      .WORD   ^X3FFF                        ;MASK FOR CMPREGMSK PSEUDO OP
                  01    006E  4574 3$:      .BYTE   ^X1
                        006F  4575          VBUSDATA        <^X19>,0              ;DO SRVC ON VBUS
            000C0000    0070  4576 4$:      .LONG   ^X000C0000                    ;TCSR.IICR CONTENTS
            80FFFFFF    0074  4577 5$:      .LONG   ^X80FFFFFF                    ;MASK FOR CMPREGMSK PSEUDO OP
                        0078  4578
                        0078  4579
                        0078  4580 BEGIN_CPU_DATA
                        0002  4581
                        0002  4582 DCS_DATA
                        0001  4583
U 00, 7700,C800,0364,0300,0470,1801     0001  4584 ;% 00    NOP
U 01, 7700,4800,5BE4,0300,1470,1802     000C  4588 ;% 01    WCTRL/INIR_WB,WB_R[TEMP0]     ;LOAD INIR REGISTER WITH ONE'S
U 02, 7700,0800,5BE4,0304,1670,1803     0017  4592 ;% 02    WCTRL/TCSR_WB,WB_R[TEMP1]     ;SET UP TCSR REGISTER
U 03, 7700,8800,5BE4,0308,1670,1804     0022  4596 ;% 03    WCTRL/TCSR_WB,WB_R[TEMP2]     ;INCREMENT IICR REGISTER
U 04, 6700,C800,0364,0300,1E70,1805     002D  4600 ;% 04    RDM_WB,WCTRL/TCSR.IICR        ;READ TCSR.IICR TO RDM
U 05, 7700,C800,5BE4,03D8,4870,1806     0038  4604 ;% 05    PC_R[ZERO]                    ;CAUSE PC TO PREFETCH
U 06, 7700,C800,0364,1700,0470,1807     0043  4608 ;% 06    IRD1TEST                      ;ALLOW INT TO HAPPEN
U 07, 7600,C800,0364,0300,0470,1808     004E  4612 ;% 07    NOP,STROBE.V.BUS              ;STROBE THE VBUS
U 08, 7300,4800,0364,0300,0470,1809     0059  4616 ;% 08    NOP,STOP.CLOCK                ;STOP THE CPU CLOCK
                        0064  4620
                        0064  4621 RTEMP_DATA
                        0001  4622
            0000FFFF    0001  4623          .LONG   ^X0000FFFF                    ;DATA FOR INIR REGISTER
            00100000    0005  4624          .LONG   ^X00100000                    ;DATA FOR TCSR REGISTER
            00200000    0009  4625          .LONG   ^X00200000                    ;STEP THE TIMER
                        000D  4626
                        000D  4627 CACHE_DATA
```

```
                     0001  4628
          00000000   0001  4629          .LONG   ^X00000000                      ;VALIDATE LOCATION 0
          00000000   0005  4630          .LONG   ^X00000000                      ;AND 4
                     0009  4631
                     0009  4632 END_CPU_DATA
                     0078  4633
                     0078  4634
                     0078  4635 END_TEST_DATA
                     0078
                     0078        ;-----------------------------------------------------------------------
                     0078  4636 ENDTEST
```

```
0022        .SBTTL  "TEST   07A     INTERVAL TIMER INTERRUPT TEST
0022  4638 ;*******************************************************************************
0022  4639 ;++
0022  4640 ;
0022  4641 ; FUNCTIONAL DESCRIPTION:
0022  4642 ;
0022  4643 ;       THIS TEST INSURES THAT AN INTERVAL TIMER INTERRUPT IS GENERATED
0022  4644 ;       WHEN THE IR BIT IS SET IN THE TCSR REGISTER.
0022  4645 ;
0022  4646 ;
0022  4647 ; LOGIC DESCRIPTION:
0022  4648 ;
0022  4649 ;       SUBTEST #1:
0022  4650 ;       1.  LOAD MICRO CODE FOR SUBTEST #1
0022  4651 ;       2.  EXECUTE DCS PROGRAM
0022  4652 ;               a.  WRITE ZEROS TO THE INIR REGISTER
0022  4653 ;               b.  SET THE T BIT IN TCSR TO CLEAR THE IICR REGISTER
0022  4654 ;               c.  LOAD INIR REGISTER WITH ONES
0022  4655 ;               d.  SET THE TVP BIT IN THE TCSR REGISTER
0022  4656 ;               e.  SET THE T BIT IN TCSR TO TRANSFER INIR TO IICR
0022  4657 ;               f.  STEP THE IICR REGISTER BY ONE TO CAUSE AN OVERFLOW
0022  4658 ;               g.  READ THE TCSR.IICR REGISTERS TO THE RDM
0022  4659 ;               h.  STROBE THE VBUS FOR TIMER INT AND INT PEND
0022  4660 ;       3.  TEST THE TCSR.IICR REGISTERS FOR CORRECT DATA
0022  4661 ;       4.  IF NO ERROR TEST THE VBUS FOR INT PEND AND TIMER INT
0022  4662 ;       5.  IF NO ERROR EXECUTE DCS PROGRAM
0022  4663 ;               a.  LOAD THE PC TO CAUSE PREFETCH
0022  4664 ;               b.  DISABLE ^P INTERRUPTS
0022  4665 ;               c.  PERFORM IRD1TEST TO ALLOW INTERRUPT TO HAPPEN
0022  4666 ;               d.  STROBE THE VBUS FOR THE MICRO VECTOR LINES
0022  4667 ;       6.  TEST THE VBUS FOR THE CORRECT MICRO VECTOR
0022  4668 ;       7.  IF NO ERROR TEST THE CS ADDRESS FOR THE CORRECT VECTOR
0022  4669 ;       8.  GO TO SUBTEST #2
0022  4670 ;
0022  4671 ;       SUBTEST #2:
0022  4672 ;       1.  LOAD MICRO CODE FOR SUBTEST #2
0022  4673 ;       2.  EXECUTE DCS PROGRAM
0022  4674 ;               a.  WRITE ZEROS TO THE INIR REGISTER
0022  4675 ;               b.  SET THE T BIT IN TCSR TO CLEAR THE IICR REGISTER
0022  4676 ;               c.  LOAD INIR REGISTER WITH ONES
0022  4677 ;               d.  SET THE TVP BIT IN THE TCSR REGISTER
0022  4678 ;               e.  SET THE T BIT IN TCSR TO TRANSFER INIR TO IICR
0022  4679 ;               f.  STEP THE IICR REGISTER BY ONE TO CAUSE AN OVERFLOW
0022  4680 ;               g.  READ THE TCSR.IICR REGISTERS TO THE RDM
0022  4681 ;               h.  BRANCH ON THE INTERRUPT PENDING SIGNAL
0022  4682 ;       3.  TEST FOR CORRECT CS ADDRESS BRANCH
0022  4683 ;       4.  IF NO ERROR GO TO NEXT TEST
0022  4684 ;
0022  4685 ;
0022  4686 ; TEST PATTERNS:
0022  4687 ;
0022  4688 ;       NONE
0022  4689 ;
0022  4690 ;
0022  4691 ; LOGIC DESCRIPTION:
```

```
                        0022  4692 ;
                        0022  4693 ;       THIS TEST SETS UP THE PROPER CONDITIONS IN THE TOK GATE ARRAY TO CAUSE
                        0022  4694 ;       AN INTERVAL TIMER INTERRUPT.  IF THE VBUS DATA FAILS, STUDY THE FAILING
                        0022  4695 ;       BIT POSITION CAREFULLY AND YOU SHOULD BE ABLE TO VERY RAPIDLY ISOLATE
                        0022  4696 ;       THE PROBLEM TO ATLEAST A MODULE.
                        0022  4697 ;       SUBTEST 2 CHECKS THAT THE DISCRETE BRANCHING LOGIC ON DPM GATES
                        0022  4698 ;       INTERRUPT PENDING ONTO CS ADDRESS BIT 01.
                        0022  4699 ;
                        0022  4700 ;
                        0022  4701 ; ASSUMPTIONS:
                        0022  4702 ;
                        0022  4703 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
                        0022  4704 ;
                        0022  4705 ;
                        0022  4706 ; ERROR DESCRIPTION:
                        0022  4707 ;
                        0022  4708 ;       SUBTEST #1
                        0022  4709 ;       FIRST IFERROR:
                        0022  4710 ;               EXPECTED TCSR.IICR CONTENTS
                        0022  4711 ;               RECEIVED TCSR.IICR CONTENTS
                        0022  4712 ;
                        0022  4713 ;       SECOND IFERROR:
                        0022  4714 ;               EXPECTED VBUS DATA
                        0022  4715 ;               RECEIVED VBUS DATA
                        0022  4716 ;                       (BITS <7:0> = VBUS BIT POSITION)
                        0022  4717 ;                       (BITS <8> = BIT STATE)
                        0022  4718 ;                       (NOTE: A FAILURE OF BIT 05, INT PEND, WOULD INDICATE
                        0022  4719 ;                               THAT ECO L0003-TW001 IS MISSING)
                        0022  4720 ;
                        0022  4721 ;       THIRD IFERROR:
                        0022  4722 ;               EXPECTED VBUS DATA
                        0022  4723 ;               RECEIVED VBUS DATA
                        0022  4724 ;                       (BITS <7:0> = VBUS BIT POSITION)
                        0022  4725 ;                       (BITS <8> = BIT STATE)
                        0022  4726 ;
                        0022  4727 ;       FORTH IFERROR:
                        0022  4728 ;               EXPECTED CS ADDRESS
                        0022  4729 ;               RECEIVED CS ADDRESS
                        0022  4730 ;
                        0022  4731 ;       SUBTEST #2:
                        0022  4732 ;               EXPECTED CS ADDRESS
                        0022  4733 ;               RECEIVED CS ADDRESS
                        0022  4734 ;
                        0022  4735 ;--
                        0022  4736 ;*******************************************************************************
                        0022  4737 ;///////////////////////////////////////////////////////////////////////////////
                        0022  4738 ;+
                        0022  4739 ;SUBTEST #1
                        0022  4740 ;
                        0022  4741 ;       TEST FOR TIMER INTERRUPT
                        0022  4742 ;-
                        0022  4743         SUBTEST                                         ;SUBTEST #1
                        0025
                        0025               ;///////////////////////////////////////////////////////////////////////
                        0025  4744         INITIALIZE                                      ;INIT THE CPU
```

```
                     0027  4745          EXPUTRAP                                ;EXPECT A U-TRAP
                     0029  4746          LOADDCS         7$,,09,1                ;LOAD MICRO CODE FOR SUBTEST
                     0030  4747          ERRLOOP                                 ;ERROR LOOP POINT
                     0032  4748          FETCH           0                       ;START DCS PROGRAM
                     0037  4749          BRSTCLK         <^X0A>                  ;END DCS PROGRAM
                     003B  4750          CMPREGMSK       WHREG,1$,,2$,,L,        ;TEST TCSR.IICR REGISTERS
                     0047  4751          IFERROR         ,<TOK>,                 ;REGISTERS NOT CORRECT
                     004D  4752          CMPVBUS         6$                      ;TEST FOR INT PEND, TIMER INT
                     0052  4753          IFERROR         ,<<TOK><INT>>,<<UBI><MIC>>;SIGNALS INCORRECT
                     005B  4754          FETCH           <^X0B>                  ;START DCS PROGRAM
                     0060  4755          BRSTCLK         <^X11>                  ;END DCS PROGRAM
                     0064  4756          CMPVBUS         3$                      ;TEST VBUS CONTENTS
                     0069  4757          IFERROR         ,<<INT><UTR>>,<<UBI><MIC>>;VBUS NOT CORRECT
                     0072  4758          CMPREGMSK       CSTRP,4$,,5$,,W,        ;TEST FOR INTERRUPT ADDRESS
                     007E  4759          IFERROR         ,<MSQ>,                 ;INCORRECT ADDRESS
                     0084  4760
                     0084  4761
                     0084  4762
                     0084  4763 ;////////////////////////////////////////////////////////////////////////////
                     0084  4764 ;+
                     0084  4765 ;SUBTEST #2
                     0084  4766 ;
                     0084  4767 ;       TEST FOR INTERRUPT PENDING BRANCH
                     0084  4768 ;-
                     0084  4769          SUBTEST                                 ;SUBTEST #2
                     0087
                     0087       ;////////////////////////////////////////////////////////////////////////////
                     0087  4770          INITIALIZE                              ;INIT THE CPU
                     0089  4771          LOADDCS         8$,,09,1                ;LOAD MICRO CODE FOR SUBTEST
                     0090  4772          ERRLOOP                                 ;ERROR LOOP POINT
                     0092  4773          FETCH           0                       ;START DCS PROGRAM
                     0097  4774          BRSTCLK         <^X0A>                  ;END DCS PROGRAM
                     009B  4775          CMPREGMSK       CSTRP,9$,,5$,,W,        ;TEST CS ADDRESS
                     00A7  4776          IFERROR                                 ;INCORRECT BRANCH
                     00AC  4777          SKIP                                    ;GO TO NEXT TEST
                     00B3  4778
                     00B3  4779
                     00B3  4780 BEGIN_TEST_DATA
                     00B3
                     00B3       ;-----------------------------------------------------------------------
                     00B3  4781
           01C2FFFF  00B3  4782 1$:     .LONG   ^X01C2FFFF                      ;TCSR.IICR REGISTER CONTENTS
           81FFFFFF  00B7  4783 2$:     .LONG   ^X81FFFFFF                      ;MASK FOR CMPREGMSK PSEUDO OP
                 04  00BB  4784 3$:     .BYTE   ^X04
                     00BC  4785          VBUSDATA        <^X08>,1                ;MICRO VECTOR LINES
                     00BD  4786          VBUSDATA        <^X09>,0
                     00BE  4787          VBUSDATA        <^X0A>,1
                     00BF  4788          VBUSDATA        <^X0B>,1
               003B  00C0  4789 4$:      .WORD   ^X003B                          ;INTERVAL TIMER INT CS ADDRESS
               3FFF  00C2  4790 5$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                 02  00C4  4791 6$:      .BYTE   ^X02
                     00C5  4792          VBUSDATA        <^X05>,0                ;INT PEND
                     00C6  4793          VBUSDATA        <^X23>,0                ;TIMER INT
                     00C7  4794 7$:                                              ;MICRO CODE SUBTEST #1
U 09, 7600,4800,0364,0300,0470,180A  00C7  4795 ;% 09    NOP,STROBE.V.BUS
```

```
                                  00D2  8$:                                                  ;MICRO CODE SUBTEST #2
U 09, 7700,4800,0364,AF00,0470,1800 00D2  ;% 09   BUT/INT-TIMSERV,NEXT/1800                  ;BUT ON INT PEND
                             1800 00DD  9$:     .WORD   ^X1800                               ;CS BRANCH ADDRESS
                                  00DF
                                  00DF
                                  00DF  BEGIN_CPU_DATA
                                  0002
                                  0002  DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801 0001  ;% 00   NOP
U 01, 7700,4800,5BE4,0318,1470,1802 000C  ;% 01   WCTRL/INIR_WB,WB_R[TEMP6]                  ;CLEAR INIR REGISTER
U 02, 7700,0800,5BE4,031C,1670,1803 0017  ;% 02   WCTRL/TCSR_WB,WB_R[TEMP7]                  ;CLEAR IICR REGISTER
U 03, 7700,4800,5BE4,0300,1470,1804 0022  ;% 03   WCTRL/INIR_WB,WB_R[TEMP0]                  ;LOAD INIR REGISTER WITH FFFF
U 04, 7700,0800,5BE4,0304,1670,1805 002D  ;% 04   WCTRL/TCSR_WB,WB_R[TEMP1]                  ;SET TVP BIT IN TCSR REGISTER
U 05, 7700,0800,5BE4,0308,1670,1806 0038  ;% 05   WCTRL/TCSR_WB,WB_R[TEMP2]                  ;TRANSFER INIR TO IICR REGISTER
U 06, 7700,C800,5BE4,030C,1670,1807 0043  ;% 06   WCTRL/TCSR_WB,WB_R[TEMP3]                  ;STEP IICR BY ONE
U 07, 6700,4800,0364,0300,1E70,1808 004E  ;% 07   RDM_WB,WCTRL/TCSR.IICR                     ;READ TCSR.IICR TO RDM
U 08, 7700,4800,0364,0300,0470,1809 0059  ;% 08   NOP
U 09, 7600,4800,0364,0300,0470,180A 0064  ;% 09   NOP,STROBE.V.BUS
U 0A, 7300,C800,0364,0300,0470,180B 006F  ;% 0A   NOP,STOP.CLOCK
U 0B, 7700,4800,0364,0300,0470,180C 007A  ;% 0B   NOP
U 0C, 7700,4800,5BE4,03D8,4870,180D 0085  ;% 0C   PC_R[ZERO]                                  ;CAUSE XB'S TO PREFETCH
U 0D, 7700,0800,5BE4,0310,2470,180E 0090  ;% 0D   WCTRL/LOADCRAR,WB_R[TEMP4]                 ;KILL ^P INTERRUPT
U 0E, 7700,4800,5BE4,0314,2C70,180F 009B  ;% 0E   WCTRL/CONWRITE,WB_R[TEMP5]                 ;...
U 0F, 7700,C800,0364,1700,0470,1810 00A6  ;% 0F   IRD1TEST                                    ;ALLOW INTERRUPT TO HAPPEN
U 10, 7600,4800,0364,0300,0470,1811 00B1  ;% 10   NOP,STROBE.V.BUS                            ;...
U 11, 7300,4800,0364,0300,0470,1812 00BC  ;% 11   NOP,STOP.CLOCK                              ;STOP THE CPU CLOCK
                                  00C7
                                  00C7  RTEMP_DATA
                                  0001
                         0000FFFF 0001          .LONG   ^X0000FFFF                          ;DATA FOR INIR REGISTER
                         01000000 0005          .LONG   ^X01000000                          ;SET TVP BIT IN TCSR
                         01500000 0009          .LONG   ^X01500000                          ;TRANSFER INIR TO IICR REGISTER
                         01640000 000D          .LONG   ^X01640000                          ;STEP IICR BY ONE
                         00C00000 0011          .LONG   ^X00C00000                          ;KILL ^P INTERRUPT
                         00400000 0015          .LONG   ^X00400000                          ;...
                         00000000 0019          .LONG   ^X00000000                          ;CLEAR INIR AND IICR REGISTERS
                         00100000 001D          .LONG   ^X00100000                          ;TRANSFER INIR TO IICR REGISTER
                                  0021
                                  0021  CACHE_DATA
                                  0001
                         00000000 0001          .LONG   ^X00000000                          ;VALIDATE CACHE ADDRESS 0
                         00000000 0005          .LONG   ^X00000000                          ;AND 4
                                  0009
                                  0009  END_CPU_DATA
                                  00DF
                                  00DF
                                  00DF  END_TEST_DATA
                                  00DF
                                  00DF        ;-----------------------------------------------------------------------
                                  00DF  ENDTEST
```

```
0023          .SBTTL  "TEST   07B     TEST ARITHMETIC TRAP INTERRUPT
0023  4906 ;*******************************************************************************
0023  4907 ;++
0023  4908 ;
0023  4909 ; FUNCTIONAL DESCRIPTION:
0023  4910 ;
0023  4911 ;       THIS TEST CHECKS THAT AN ARITHMETIC TRAP CAN BE GENERATED AND HANDLED
0023  4912 ;       BY THE CPU.
0023  4913 ;
0023  4914 ;
0023  4915 ; TEST ALGORITHM:
0023  4916 ;
0023  4917 ;       1.  EXECUTE DCS PROGRAM
0023  4918 ;               a.  DISABLE ANY CONTROL P INTERRUPT
0023  4919 ;               b.  LOAD PC WITH ZERO TO CAUSE PREFETCH
0023  4920 ;               c.  PERFORM IRD1TEST TO LOAD IR WITH AN ARITHMETIC INSTRUCTION
0023  4921 ;               d.  SET THE IV BIT IN THE PSL REGISTER
0023  4922 ;               e.  SET THE V BIT IN THE PSW REGISTER
0023  4923 ;               f.  PERFORM AN IRD1TEST TO ALLOW THE INTERRUPT TO HAPPEN
0023  4924 ;       2.  TEST THE VBUS FOR DO SRVC
0023  4925 ;       3.  IF NO ERROR TEST THE CS ADDRESS FOR THE CORRECT INTERRUPT VECTOR
0023  4926 ;       4.  IF NO ERROR EXECUTE DCS PROGRAM
0023  4927 ;               a.  READ THE ATCR REGISTER TO THE RDM
0023  4928 ;       5.  TEST THE ATCR REGISTER FOR CORRECT DATA
0023  4929 ;       6.  IF NO ERROR GO TO THE NEXT TEST
0023  4930 ;
0023  4931 ;
0023  4932 ; TEST PATTERNS:
0023  4933 ;
0023  4934 ;       NONE
0023  4935 ;
0023  4936 ;
0023  4937 ; LOGIC DESCRIPTION:
0023  4938 ;
0023  4939 ;       THIS TEST SETS UP THE PROPER CONDITIONS IN THE CCC GATE ARRAY TO
0023  4940 ;       GENERATE AN ARITHMETIC TRAP.  THE SAC GATE ARRAY IS RESPONSIBLE FOR
0023  4941 ;       HANDLING THE INTERRUPT AND GENERATING THE PROPER INTERRUPT VECTOR.
0023  4942 ;
0023  4943 ;
0023  4944 ; ASSUMPTIONS:
0023  4945 ;
0023  4946 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
0023  4947 ;
0023  4948 ;
0023  4949 ; ERROR DESCRIPTION:
0023  4950 ;
0023  4951 ;       FIRST IFERROR:
0023  4952 ;               EXPECTED VBUS DATA
0023  4953 ;               RECEIVED VBUS DATA
0023  4954 ;                       (BITS <7:0> = VBUS BIT POSITION)
0023  4955 ;                       (BITS <8> = BIT STATE)
0023  4956 ;
0023  4957 ;       SECOND IFERROR:
0023  4958 ;               EXPECTED CS ADDRESS
0023  4959 ;               RECEIVED CS ADDRESS
```

```
                        0023  4960 ;
                        0023  4961 ;       THIRD IFERROR:
                        0023  4962 ;               EXPECTED ATCR REGISTER CONTENTS
                        0023  4963 ;               RECEIVED ATCR REGISTER CONTENTS
                        0023  4964 ;
                        0023  4965 ;--
                        0023  4966 ;*******************************************************************************
                        0023  4967
                        0023  4968
                        0023  4969         INITIALIZE                              ;INIT THE CPU
                        0025  4970         EXPUTRAP                                ;EXPECT A TRAP
                        0027  4971         ERRLOOP                                 ;ERROR LOOP POINT
                        0029  4972         FETCH           0                       ;START DCS PROGRAM
                        002E  4973         BRSTCLK         9                       ;END DCS PROGRAM
                        0032  4974         CMPVBUS         1$                      ;TEST FOR DO SRVC
                        0037  4975         IFERROR         ,<<CCC><SAC>>,          ;DO SRVC NOT PRESENT
                        003E  4976         CMPREGMSK       CSTRP,2$,,3$,,W,        ;TEST FOR CORRECT TRAP ADDRESS
                        004A  4977         IFERROR         ,<<SAC><MSQ>>,          ;NOT CORRECT TRAP
                        0051  4978         FETCH           <^X0A>                  ;START DCS PROGRAM
                        0056  4979         BRSTCLK         <^X0C>                  ;END DCS PROGRAM
                        005A  4980         CMPREGMSK       WHREG,4$,,5$,,L,        ;TEST ATCR REGISTER CONTENTS
                        0066  4981         IFERROR         ,<CCC>,                 ;REGISTER NOT SET RIGHT
                        006C  4982         SKIP                                    ;GO TO NEXT TEST
                        0073  4983
                        0073  4984
                        0073  4985 BEGIN_TEST_DATA
                        0073
                        0073       ;-------------------------------------------------------------------------
                        0073  4986
                     01 0073  4987 1$:      .BYTE   ^X01
                        0074  4988          VBUSDATA        <^X19>,0                ;DO SRVC ON VBUS
                   0011 0075  4989 2$:      .WORD   ^X0011                          ;ARITHMETIC TRAP
                   3FFF 0077  4990 3$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
               00000001 0079  4991 4$:      .LONG   ^X00000001                      ;INTERGER OVERFLOW
               0000000F 007D  4992 5$:      .LONG   ^X0000000F                      ;MASK FOR CMPREGMSK PSEUDO OP
                        0081  4993
                        0081  4994
                        0081  4995
                        0081  4996 BEGIN_CPU_DATA
                        0002  4997
                        0002  4998 DCS_DATA
                        0001  4999
U 00, 7700,C800,0364,0300,0470,1801  0001  5000 ;% 00   NOP
U 01, 7700,4800,5BE4,0300,2470,1802  000C  5004 ;% 01   WCTRL/LOADCRAR,WB_R[TEMP0]          ;KILL ^P INTERRUPT
U 02, 7700,0800,5BE4,0304,2C70,1803  0017  5008 ;% 02   WCTRL/CONWRITE,WB_R[TEMP1]         ;...
U 03, 7700,4800,5BE4,03D8,4870,1804  0022  5012 ;% 03   PC_R[ZERO]                        ;CAUSE XB'S TO PREFETCH
U 04, 7700,4800,0364,1700,0470,1805  002D  5016 ;% 04   IRD1TEST                         ;LOAD IR WITH INST
U 05, 7700,C800,5BF4,0308,0070,1806  0038  5020 ;% 05   PSL_R[TEMP2]                     ;SET IV BIT IN PSL
U 06, 7700,4800,5BE4,030C,0A70,1807  0043  5024 ;% 06   CCPSL/CC_WB.CCBR_ALUS,WB_R[TEMP3]  ;SET V BIT IN PSW
U 07, 7700,C800,0364,1700,0470,1808  004E  5028 ;% 07   IRD1TEST                         ;LOOK FOR INTERRUPT
U 08, 7600,4800,0364,0300,0470,1809  0059  5032 ;% 08   NOP,STROBE.V.BUS                 ;...
U 09, 7300,4800,0364,0300,0470,180A  0064  5036 ;% 09   NOP,STOP.CLOCK                   ;STOP THE CPU CLOCK
U 0A, 7700,C800,0364,0300,0470,180B  006F  5040 ;% 0A   NOP
U 0B, 6700,4800,0364,0300,0E70,180C  007A  5044 ;% 0B   RDM_WB,CCMISC/WB_ATCR.CCBR_SIGND  ;READ ATCR TO RDM
U 0C, 7300,C800,0364,0300,0470,180D  0085  5048 ;% 0C   NOP,STOP.CLOCK
```

```
                        0090  5052
                        0090  5053 RTEMP_DATA
                        0001  5054
                00C00000 0001  5055          .LONG   ^X00C00000                  ;KILL ^P INTERRUPT
                00400000 0005  5056          .LONG   ^X00400000                  ;...
                00000020 0009  5057          .LONG   ^X00000020                  ;SET PSL IV BIT
                00000002 000D  5058          .LONG   ^X00000002                  ;SET PSW V BIT
                        0011  5059
                        0011  5060 CACHE_DATA
                        0001  5061
                00000048 0001  5062          .LONG   ^X00000048                  ;VALIDATE CACHE LOCATION 0
                00000000 0005  5063          .LONG   ^X00000000                  ;AND 4
                        0009  5064
                        0009  5065 END_CPU_DATA
                        0081  5066
                        0081  5067
                        0081  5068 END_TEST_DATA
                        0081
                        0081          ;-------------------------------------------------------------------------
                        0081  5069 ENDTEST
```

```
                        001E        .SBTTL  "TEST   07C     TEST POWER FAIL INTERRUPT
                        001E  5071 ;*******************************************************************************
                        001E  5072 ;++
                        001E  5073 ;
                        001E  5074 ; FUNCTIONAL DESCRIPTION:
                        001E  5075 ;
                        001E  5076 ;       THIS TEST CHECKS THE OPERATION OF THE POWER FAIL INTERRUPT.
                        001E  5077 ;
                        001E  5078 ;
                        001E  5079 ; TEST ALGORITHM:
                        001E  5080 ;
                        001E  5081 ;       1.  LOAD FIRST MICRO INSTRUCTION
                        001E  5082 ;       2.  SET ACLO FROM THE RDM
                        001E  5083 ;       3.  EXECUTE DCS PROGRAM
                        001E  5084 ;               a.  PERFORM RETURN AND SURPRESS BUS CYCLE TO GATE ACLO
                        001E  5085 ;               b.  DISABLE ^P INTERRUPTS
                        001E  5086 ;               c.  LOAD PC TO CAUSE PREFETCH TO HAPPEN
                        001E  5087 ;               d.  PERFORM IRD1TEST TO ALLOW INTERRUPT TO HAPPEN
                        001E  5088 ;       4.  CLEAR ACLO FROM RDM
                        001E  5089 ;       5.  TEST VBUS FOR INTERRUPT PENDING
                        001E  5090 ;       6.  IF NO ERROR TEST CS ADDRESS FOR CORRECT BRANCH
                        001E  5091 ;       7.  IF NO ERROR GO TO NEXT TEST
                        001E  5092 ;
                        001E  5093 ;
                        001E  5094 ; TEST PATTERNS:
                        001E  5095 ;
                        001E  5096 ;       NONE
                        001E  5097 ;
                        001E  5098 ;
                        001E  5099 ; LOGIC DESCRIPTION:
                        001E  5100 ;
                        001E  5101 ;       MOST OF THE GATING FOR ACLO IS DONE ON UBI.  THE INT GATE ARRAY IS
                        001E  5102 ;       RESPONSIBLE FOR RECOGNIZING POWER FAIL AND ASSERTING INTERRUPT PENDING.
                        001E  5103 ;       ANY ERRORS SHOULD BE CONFINED TO UBI.
                        001E  5104 ;
                        001E  5105 ;
                        001E  5106 ; ASSUMPTIONS:
                        001E  5107 ;
                        001E  5108 ;       THIS TEST ASSUMES THE PREVIOUS INTERRUPT TESTS HAVE RUN SUCCESSFULLY.
                        001E  5109 ;
                        001E  5110 ;
                        001E  5111 ; ERROR DESCRIPTION:
                        001E  5112 ;
                        001E  5113 ;       FIRST IFERROR
                        001E  5114 ;               EXPECTED VBUS DATA (INTERRUPT PENDING)
                        001E  5115 ;               RECEIVED VBUS DATA
                        001E  5116 ;                       (BITS <7:0> = VBUS BIT POSITION)
                        001E  5117 ;                       (BITS <8> = BIT STATE)
                        001E  5118 ;
                        001E  5119 ;       SECOND IFERROR
                        001E  5120 ;               EXPECTED CS ADDRESS
                        001E  5121 ;               RECEIVED CS ADDRESS
                        001E  5122 ;
                        001E  5123 ;--
                        001E  5124 ;*******************************************************************************
```

```
                                001E
                                001E
                                001E
                                001E          INITIALIZE                                ;INIT THE CPU
                                0020          EXPUTRAP                                  ;EXPECT A MICRO TRAP
                                0022          ERRLOOP                                   ;ERROR LOOP POINT
                                0024          FETCH          0                          ;START DCS PROGRAM
                                0029          LOADREG        RDCTRL,1$,,B               ;SET ACLO
                                0031          BRSTCLK        7                          ;END DCS PROGRAM
                                0035          LOADREG        RDCTRL,5$,,B               ;CLEAR ACLO
                                003D          CMPVBUS        2$                         ;TEST FOR INT PEND
                                0042          IFERROR        ,<INT>,<UBI>               ;NO INT PENDING
                                0049          CMPREGMSK      CSTRP,3$,,4$,,W,           ;TEST FOR CORRECT TRAP ADDRESS
                                0055          IFERROR        ,<INT>,<UBI>               ;INCORRECT ADDRESS
                                005C          SKIP                                      ;GO TO NEXT TEST
                                0063
                                0063
                                0063  BEGIN_TEST_DATA
                                0063
                                0063      ;-----------------------------------------------------------------------
                                0063
                          A8    0063  1$:     .BYTE   ^XA8                              ;SET ACLO
                          01    0064  2$:     .BYTE   ^X01
                                0065          VBUSDATA       <^X05>,0                   ;INT PEND ON VBUS
                        003F    0066  3$:     .WORD   ^X003F                            ;POWER FAIL TRAP ADDRESS
                        3FFF    0068  4$:     .WORD   ^X3FFF                            ;MASK FOR CMPREGMSK PSEUDO OP
                          28    006A  5$:     .BYTE   ^X28                              ;CLEAR ACLO
                                006B
                                006B
                                006B  BEGIN_CPU_DATA
                                0002
                                0002  DCS_DATA
                                0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 7700,4D80,0364,0300,0470,1802  000C  ;% 01   MISC/RSBC                                ;GATE ACLO TO INTERRUPT
U 02, 7700,C800,5BE4,0300,2470,1803  0017  ;% 02   WCTRL/LOADCRAR,WB_R[TEMP0]               ;DISABLE ^P INTERRUPTS
U 03, 7700,8800,5BE4,0304,2C70,1804  0022  ;% 03   WCTRL/CONWRITE,WB_R[TEMP1]               ;...
U 04, 7700,C800,5BE4,03D8,4870,1805  002D  ;% 04   PC_R[ZERO]                               ;CAUSE PREFETCH
U 05, 7600,4800,0364,1700,0470,1806  0038  ;% 05   IRD1TEST,STROBE.V.BUS                    ;ALLOW INTERRUPT TO HAPPEN
U 06, 7700,C800,0364,0300,0470,1807  0043  ;% 06   NOP                                      ;...
U 07, 7300,C800,0364,0300,0470,1808  004E  ;% 07   NOP,STOP.CLOCK                           ;STOP THE CPU CLOCK
                                0059
                                0059  RTEMP_DATA
                                0001
                    00C00000    0001          .LONG   ^X00C00000                        ;DISABLE ^P INTERRUPTS
                    00400000    0005          .LONG   ^X00400000                        ;...
                                0009
                                0009  CACHE_DATA
                                0001
                    00000000    0001          .LONG   ^X00000000
                    00000000    0005          .LONG   ^X00000000
                                0009
                                0009  END_CPU_DATA
                                006B
                                006B
```

```
                        006B  5202 END_TEST_DATA
                        006B
                        006B        ;-----------------------------------------------------------------------------
                        006B  5203 ENDTEST
```

```
0015        .SBTTL  "TEST   07D     REI CHECK TEST 1
0015  5205 ;*******************************************************************************
0015  5206 ;++
0015  5207 ;
0015  5208 ; FUNCTIONAL DESCRIPTION:
0015  5209 ;
0015  5210 ;       THIS TEST IS ONE IN A SERIES OF FIVE THAT VERIFIES THE OPERATION
0015  5211 ;       OF REI CHECK.
0015  5212 ;
0015  5213 ;
0015  5214 ; TEST ALGORITHM:
0015  5215 ;
0015  5216 ;       1.  LOAD MICRO CODE FOR THIS SUBTEST
0015  5217 ;       2.  CREATE A TEST LOOP OF 4 TO SELECT PSL'S
0015  5218 ;       3.  LOAD RDM WDREG WITH TEST PSL
0015  5219 ;       4.  EXECUTE DCS PROGRAM
0015  5220 ;               a.  SET AST REGISTER ACCORDING TO SUBTEST
0015  5221 ;               b.  PERFORM REI CHECK
0015  5222 ;       5.  TEST VBUS FOR UVCTR BRANCH (SUBTEST #1)
0015  5223 ;       6.  IF NO ERROR TEST CS ADDRESS FOR CORRECT VECTOR
0015  5224 ;       7.  IF NO ERROR GO TO STEP 3 FOR REMAINING PSL'S
0015  5225 ;
0015  5226 ;
0015  5227 ; TEST PATTERNS:
0015  5228 ;
0015  5229 ;       SUBTEST #1: (ASTLVL = 0)
0015  5230 ;               ITERATION       PSL     CS ADDRESS
0015  5231 ;                   1            0         1809
0015  5232 ;                   2            1         1809
0015  5233 ;                   3            2         1809
0015  5234 ;                   4            3         1809
0015  5235 ;
0015  5236 ;       SUBTEST #2: (ASTLVL = 1)
0015  5237 ;               ITERATION       PSL     CS ADDRESS
0015  5238 ;                   1            0         1808
0015  5239 ;                   2            1         1809
0015  5240 ;                   3            2         1809
0015  5241 ;                   4            3         1809
0015  5242 ;
0015  5243 ;       SUBTEST #3: (ASTLVL = 2)
0015  5244 ;               ITERATION       PSL     CS ADDRESS
0015  5245 ;                   1            0         1808
0015  5246 ;                   2            1         1808
0015  5247 ;                   3            2         1809
0015  5248 ;                   4            3         1809
0015  5249 ;
0015  5250 ;       SUBTEST #4: (ASTLVL = 2)
0015  5251 ;               ITERATION       PSL     CS ADDRESS
0015  5252 ;                   1            0         1808
0015  5253 ;                   2            1         1808
0015  5254 ;                   3            2         1808
0015  5255 ;                   4            3         1809
0015  5256 ;
0015  5257 ;
0015  5258 ; LOGIC DESCRIPTION:
```

```
                        0015  5259 ;
                        0015  5260 ;       THIS TEST CHECKS THAT THE INT GATE ARRAY CAN PERFORM AN REI CHECK
                        0015  5261 ;       MICRO ORDER.  WHEN YOU PERFORM THIS MICRO ORDER YOU MUST HAVE THE
                        0015  5262 ;       TEST PSL ENABLED ONTO THE WBUS SO INT CAN DO THE COMPARISON.  IN
                        0015  5263 ;       THIS TEST I'M COMPARING THE AST LEVEL AGAINST THE CURRENT MODE.
                        0015  5264 ;       ANY FAILURES SHOULD BE CONFINED TO THE INT GATE ARRAY ON THE UBI
                        0015  5265 ;       MODULE.
                        0015  5266 ;
                        0015  5267 ;
                        0015  5268 ; ASSUMPTIONS:
                        0015  5269 ;
                        0015  5270 ;       THIS TEST ASSUMES WBUS IS OPERATIONAL.
                        0015  5271 ;
                        0015  5272 ;
                        0015  5273 ; ERROR DESCRIPTION:
                        0015  5274 ;
                        0015  5275 ;       SUBTEST #1:
                        0015  5276 ;               FIRST IFERROR:
                        0015  5277 ;                       EXPECTED VBUS DATA
                        0015  5278 ;                       RECEIVED VBUS DATA
                        0015  5279 ;                               (BITS <7:0> = VBUS BIT POSITION)
                        0015  5280 ;                               (BITS <8> = BIT STATE)
                        0015  5281 ;                       LOOP COUNT I
                        0015  5282 ;
                        0015  5283 ;               SECOND IFERROR:
                        0015  5284 ;                       EXPECTED CS ADDRESS
                        0015  5285 ;                       RECEIVED CS ADDRESS
                        0015  5286 ;                       LOOP COUNT I
                        0015  5287 ;
                        0015  5288 ;       SUBTEST #2,3,4:
                        0015  5289 ;               EXPECTED CS ADDRESS
                        0015  5290 ;               RECEIVED CS ADDRESS
                        0015  5291 ;               LOOP COUNT I
                        0015  5292 ;
                        0015  5293 ;--
                        0015  5294 ;*******************************************************************************
                        0015  5295 ;///////////////////////////////////////////////////////////////////////////////
                        0015  5296 ;+
                        0015  5297 ;SUBTEST #1
                        0015  5298 ;
                        0015  5299 ;       ASTLVL EQUAL 0
                        0015  5300 ;-
                        0015  5301         SUBTEST                                         ;SUBTEST #1
                        0018
                        0018       ;////////////////////////////////////////////////////////////////////////////
                        0018  5302         INITIALIZE                                      ;INIT THE CPU
                        001A  5303         EXPUTRAP                                        ;POSSIBLE FAILURE MODE
                        001C  5304         LOADDCS         4$,,01,1                        ;MODIFY UCODE FOR SUBTEST
                        0023  5305         LOOP            I,1,4                           ;CREATE A TEST LOOP OF 4
                        002A  5306         ERRLOOP                                         ;ERROR LOOP POINT
                        002C  5307         LOADREG         WDREG,5$,I,L                    ;LOAD PSL INTO WDREG
                        0034  5308         FETCH           0                               ;START DCS PROGRAM
                        0039  5309         BRSTCLK         3                               ;END DCS PROGRAM
                        003D  5310         CMPVBUS         3$                              ;TEST UVCTR BRANCH SIGNAL
                        0042  5311         IFERROR                                         ;SIGNAL NOT CORRECT
```

```
                0047  5312          CMPREGMSK       CSTRP,1$,,2$,,W,        ;TEST FOR CORRECT CS ADDRESS
                0053  5313          IFERROR         ,<INT>,<UBI>            ;CS ADDRESS INCORRECT
                005A  5314          ENDLOOP         I                       ;END TEST LOOP
                005D  5315
                005D  5316
                005D  5317  ;//////////////////////////////////////////////////////////////////////////////
                005D  5318  ;+
                005D  5319  ;SUBTEST #2
                005D  5320  ;
                005D  5321  ;       ASTLVL EQUAL 1
                005D  5322  ;-
                005D  5323          SUBTEST                                 ;SUBTEST #2
                0060
                0060        ;//////////////////////////////////////////////////////////////////////////////
                0060  5324          INITIALIZE                              ;INIT THE CPU
                0062  5325          EXPUTRAP                                ;POSSIBLE FAILURE MODE
                0064  5326          LOADDCS         6$,,01,1                ;MODIFY UCODE FOR SUBTEST
                006B  5327          LOOP            I,1,4                   ;CREATE A TEST LOOP OF 4
                0072  5328          ERRLOOP                                 ;ERROR LOOP POINT
                0074  5329          LOADREG         WDREG,5$,I,L            ;LOAD PSL INTO WDREG
                007C  5330          FETCH           0                       ;START DCS PROGRAM
                0081  5331          BRSTCLK         3                       ;END DCS PROGRAM
                0085  5332          CMPREGMSK       CSTRP,7$,I,2$,,W,       ;TEST FOR CORRECT CS ADDRESS
                0091  5333          IFERROR         ,<INT>,<UBI>            ;CS ADDRESS INCORRECT
                0098  5334          ENDLOOP         I                       ;END TEST LOOP
                009B  5335
                009B  5336  ;//////////////////////////////////////////////////////////////////////////////
                009B  5337  ;+
                009B  5338  ;SUBTEST #3
                009B  5339  ;
                009B  5340  ;       ASTLVL EQUAL 2
                009B  5341  ;-
                009B  5342          SUBTEST                                 ;SUBTEST #3
                009E
                009E        ;//////////////////////////////////////////////////////////////////////////////
                009E  5343          INITIALIZE                              ;INIT THE CPU
                00A0  5344          EXPUTRAP                                ;POSSIBLE FAILURE MODE
                00A2  5345          LOADDCS         8$,,01,1                ;MODIFY UCODE FOR SUBTEST
                00A9  5346          LOOP            I,1,4                   ;CREATE A TEST LOOP OF 4
                00B0  5347          ERRLOOP                                 ;ERROR LOOP POINT
                00B2  5348          LOADREG         WDREG,5$,I,L            ;LOAD PSL INTO WDREG
                00BA  5349          FETCH           0                       ;START DCS PROGRAM
                00BF  5350          BRSTCLK         3                       ;END DCS PROGRAM
                00C3  5351          CMPREGMSK       CSTRP,9$,I,2$,,W,       ;TEST FOR CORRECT CS ADDRESS
                00CF  5352          IFERROR         ,<INT>,<UBI>            ;CS ADDRESS INCORRECT
                00D6  5353          ENDLOOP         I                       ;END TEST LOOP
                00D9  5354
                00D9  5355
                00D9  5356  ;//////////////////////////////////////////////////////////////////////////////
                00D9  5357  ;+
                00D9  5358  ;SUBTEST #4
                00D9  5359  ;
                00D9  5360  ;       ASTLVL EQUAL 3
                00D9  5361  ;-
                00D9  5362          SUBTEST                                 ;SUBTEST #4
```

```
                                  00DC
                                  00DC        ;////////////////////////////////////////////////////////////////////////////
                                  00DC  5363          INITIALIZE                                  ;INIT THE CPU
                                  00DE  5364          EXPUTRAP                                    ;POSSIBLE FAILURE MODE
                                  00E0  5365          LOADDCS         10$,,01,1                   ;MODIFY UCODE FOR SUBTEST
                                  00E7  5366          LOOP            I,1,4                       ;CREATE A TEST LOOP OF 4
                                  00EE  5367          ERRLOOP                                     ;ERROR LOOP POINT
                                  00F0  5368          LOADREG         WDREG,5$,I,L                ;LOAD PSL INTO WDREG
                                  00F8  5369          FETCH           0                           ;START DCS PROGRAM
                                  00FD  5370          BRSTCLK         3                           ;END DCS PROGRAM
                                  0101  5371          CMPREGMSK       CSTRP,11$,I,2$,,W,          ;TEST FOR CORRECT CS ADDRESS
                                  010D  5372          IFERROR         ,<INT>,<UBI>                ;CS ADDRESS INCORRECT
                                  0114  5373          ENDLOOP         I                           ;END TEST LOOP
                                  0117  5374          SKIP                                        ;GO TO NEXT TEST
                                  011E  5375
                                  011E  5376
                                  011E  5377 BEGIN_TEST_DATA
                                  011E
                                  011E        ;-----------------------------------------------------------------------
                                  011E  5378
                            1809  011E  5379 1$:      .WORD   ^X1809                              ;RESULTS SUBTEST #1
                            3FFF  0120  5380 2$:      .WORD   ^X3FFF                              ;MASK FOR CMPREGMSK PSEUDO OP
                              01  0122  5381 3$:      .BYTE   ^X01
                                  0123  5382          VBUSDATA        <^X1B>,1                    ;UVCTR BRANCH ON VBUS
                                  0124  5383 4$:
U 01, 7700,1800,CB70,0300,7070,1802  0124  5384 ;% 01   ASTLVL_[0]                                 ;UCODE FOR SUBTEST #1
                        00000000  012F  5388 5$:      .LONG   ^X00000000                          ;PSL TABLE
                        01400000  0133  5389          .LONG   ^X01400000
                        02800000  0137  5390          .LONG   ^X02800000
                        03C00000  013B  5391          .LONG   ^X03C00000
                                  013F  5392 6$:
U 01, 7700,5800,CB70,0300,F070,1802  013F  5393 ;% 01   ASTLVL_[1]                                 ;UCODE FOR SUBTEST #2
                            1808  014A  5397 7$:      .WORD   ^X1808                              ;RESULTS SUBTEST #2
                            1809  014C  5398          .WORD   ^X1809
                            1809  014E  5399          .WORD   ^X1809
                            1809  0150  5400          .WORD   ^X1809
                                  0152  5401 8$:
U 01, 7700,5800,CB70,0301,7070,1802  0152  5402 ;% 01   ASTLVL_[2]                                 ;UCODE FOR SUBTEST #3
                            1808  015D  5406 9$:      .WORD   ^X1808                              ;RESULTS SUBTEST #3
                            1808  015F  5407          .WORD   ^X1808
                            1809  0161  5408          .WORD   ^X1809
                            1809  0163  5409          .WORD   ^X1809
                                  0165  5410 10$:
U 01, 7700,1800,CB70,0301,F070,1802  0165  5411 ;% 01   ASTLVL_[3]                                 ;UCODE FOR SUBTEST #4
                            1808  0170  5415 11$:     .WORD   ^X1808                              ;RESULTS SUBTEST #4
                            1808  0172  5416          .WORD   ^X1808
                            1808  0174  5417          .WORD   ^X1808
                            1809  0176  5418          .WORD   ^X1809
                                  0178  5419
                                  0178  5420
                                  0178  5421 BEGIN_CPU_DATA
                                  0002  5422
                                  0002  5423 DCS_DATA
                                  0001  5424
U 00, 7700,C800,0364,0300,0470,1801  0001  5425 ;% 00   NOP
```

```
ZZ-ECKAB-9.2    "TEST 07D REI CHECK TEST 1                    13-MAY-1987       Fiche 3 Frame H13          Sequence 575
ECKAB                        VAX-11/750 DPM MICRO DIAGNOSTICS     11-DEC-1985 09:34:06  VAX/VMS Macro V04-00        Page 11
V09.01                       "TEST 07D REI CHECK TEST 1           11-DEC-1985 09:33:26  [VAX750.DPM]ECKAB3.MAR;1          (1

U 01, 7700,1800,CB70,0300,7070,1802  000C  5429 ;% 01   ASTLVL_[0]                                  ;SETUP THE ASTLVL REGISTER
U 02, 5600,C800,0364,7B00,6E90,9808  0017  5433 ;% 02   WB_RDM,REI CHECK?,NEXT/1808,STROBE.V.BUS;PERFORM REI CHECK
U 03, 7300,C800,0364,0300,0470,1804  0022  5437 ;% 03   NOP,STOP.CLOCK
                                     002D  5441
                                     002D  5442 END_CPU_DATA
                                     0178  5443
                                     0178  5444
                                     0178  5445 END_TEST_DATA
                                     0178
                                     0178       ;-------------------------------------------------------------------------
                                     0178  5446 ENDTEST
```

```
0015        .SBTTL  "TEST   07E     REI CHECK TEST 2
0015  5448 ;*******************************************************************************
0015  5449 ;++
0015  5450 ;
0015  5451 ; FUNCTIONAL DESCRIPTION:
0015  5452 ;
0015  5453 ;       THIS TEST IS ONE IN A SERIES OF FIVE THAT VERIFIES THE OPERATION
0015  5454 ;       OF REI CHECK.
0015  5455 ;
0015  5456 ;
0015  5457 ; TEST ALGORITHM:
0015  5458 ;
0015  5459 ;       1.  LOAD MICRO CODE FOR THIS SUBTEST
0015  5460 ;       2.  CREATE A TEST LOOP OF 4
0015  5461 ;       3.  LOAD PSL INTO WDREG
0015  5462 ;       4.  EXECUTE DCS PROGRAM
0015  5463 ;               a.  SET CURRENT MODE ACCORDING TO SUBTEST
0015  5464 ;               b.  INVALIDATE AST REGISTER BY SETTING TO 4
0015  5465 ;               c.  PUT PSL ON WBUS WHILE PERFORMING REI CHECK
0015  5466 ;       5.  TEST CS ADDRESS FOR CORRECT ADDRESS
0015  5467 ;       6.  IF NO ERROR GO TO STEP 3 FOR REMAINING PSL'S
0015  5468 ;
0015  5469 ;
0015  5470 ; TEST PATTERNS:
0015  5471 ;
0015  5472 ;       SUBTEST #1: (CURRENT MODE = 3)
0015  5473 ;               ITERATION       PSL CURRENT MODE        CS ADDRESS
0015  5474 ;                   1                   3                   1808
0015  5475 ;                   2                   2                   180A
0015  5476 ;                   3                   1                   180A
0015  5477 ;                   4                   0                   180A
0015  5478 ;
0015  5479 ;       SUBTEST #2: (CURRENT MODE = 2)
0015  5480 ;               ITERATION       PSL CURRENT MODE        CS ADDRESS
0015  5481 ;                   1                   3                   1808
0015  5482 ;                   2                   2                   1808
0015  5483 ;                   3                   1                   180A
0015  5484 ;                   4                   0                   180A
0015  5485 ;
0015  5486 ;       SUBTEST #3: (CURRENT MODE = 1)
0015  5487 ;               ITERATION       PSL CURRENT MODE        CS ADDRESS
0015  5488 ;                   1                   3                   1808
0015  5489 ;                   2                   2                   1808
0015  5490 ;                   3                   1                   1808
0015  5491 ;                   4                   0                   180A
0015  5492 ;
0015  5493 ;       SUBTEST #4: (CURRENT MODE = 0)
0015  5494 ;               ITERATION       PSL CURRENT MODE        CS ADDRESS
0015  5495 ;                   1                   3                   1808
0015  5496 ;                   2                   2                   1808
0015  5497 ;                   3                   1                   1808
0015  5498 ;                   4                   0                   1808
0015  5499 ;
0015  5500 ;
0015  5501 ; LOGIC DESCRIPTION:
```

```
0015  5502 ;
0015  5503 ;       THIS TEST CHECKS THAT THE INT GATE ARRAY CAN PERFORM AN REI CHECK
0015  5504 ;       MICRO ORDER.  THE SAVED PSL CURRENT MODE (ON WBUS) IS COMPARED
0015  5505 ;       AGAINST THE CURRENT MODE IN THE INT GATE ARRAY. ANY ERRORS SHOULD BE
0015  5506 ;       CONFINED TO THE INT GATE ARRAY.
0015  5507 ;
0015  5508 ;
0015  5509 ; ASSUMPTIONS:
0015  5510 ;
0015  5511 ;       THIS TEST ASSUMES THE WBUS IS OPERATIONAL.
0015  5512 ;
0015  5513 ;
0015  5514 ; ERROR DESCRIPTION:
0015  5515 ;
0015  5516 ;       EXPECTED CS ADDRESS
0015  5517 ;       RECEIVED CS ADDRESS
0015  5518 ;       LOOP COUNT I
0015  5519 ;
0015  5520 ;--
0015  5521 ;*******************************************************************************
0015  5522 ;///////////////////////////////////////////////////////////////////////////////
0015  5523 ;+
0015  5524 ;SUBTEST #1
0015  5525 ;
0015  5526 ;       CURRENT MODE EQUAL 3
0015  5527 ;-
0015  5528         SUBTEST                                         ;SUBTEST #1
0018
0018        ;///////////////////////////////////////////////////////////////////////////
0018  5529         INITIALIZE                                      ;INIT THE CPU
001A  5530         EXPUTRAP                                        ;POSSIBLE FAILING MODE
001C  5531         LOADDCS         1$,,01,1                        ;MODIFY DCS FOR THIS SUBTEST
0023  5532         LOOP            I,1,4                           ;CREATE A TEST LOOP OF 4
002A  5533         LOADREG         WDREG,2$,I,L                    ;LOAD PSL INTO WDREG
0032  5534         ERRLOOP                                         ;ERROR LOOP POINT
0034  5535         FETCH           0                               ;START DCS PROGRAM
0039  5536         BRSTCLK         4                               ;END DCS PROGRAM
003D  5537         CMPREGMSK       CSTRP,3$,I,4$,,W,               ;TEST FOR CORRECT CS ADDRESS
0049  5538         IFERROR         ,<INT>,<UBI>                    ;CS ADDRESS INCORRECT
0050  5539         ENDLOOP         I                               ;END TEST LOOP
0053  5540
0053  5541
0053  5542 ;///////////////////////////////////////////////////////////////////////////////
0053  5543 ;+
0053  5544 ;SUBTEST #2
0053  5545 ;
0053  5546 ;       CURRENT MODE EQUAL 2
0053  5547 ;-
0053  5548         SUBTEST                                         ;SUBTEST #2
0056
0056        ;///////////////////////////////////////////////////////////////////////////
0056  5549         INITIALIZE                                      ;INIT THE CPU
0058  5550         EXPUTRAP                                        ;POSSIBLE FAILING MODE
005A  5551         LOADDCS         5$,,01,1                        ;MODIFY DCS FOR THIS SUBTEST
0061  5552         LOOP            I,1,4                           ;CREATE A TEST LOOP OF 4
```

```
                     0068  5553          LOADREG         WDREG,2$,I,L                    ;LOAD PSL INTO WDREG
                     0070  5554          ERRLOOP                                         ;ERROR LOOP POINT
                     0072  5555          FETCH           0                               ;START DCS PROGRAM
                     0077  5556          BRSTCLK         4                               ;END DCS PROGRAM
                     007B  5557          CMPREGMSK       CSTRP,6$,I,4$,,W,               ;TEST FOR CORRECT CS ADDRESS
                     0087  5558          IFERROR         ,<INT>,<UBI>                    ;CS ADDRESS INCORRECT
                     008E  5559          ENDLOOP         I                               ;END TEST LOOP
                     0091  5560
                     0091  5561
                     0091  5562 ;////////////////////////////////////////////////////////////////////////////////
                     0091  5563 ;+
                     0091  5564 ;SUBTEST #3
                     0091  5565 ;
                     0091  5566 ;       CURRENT MODE EQUAL 1
                     0091  5567 ;-
                     0091  5568          SUBTEST                                         ;SUBTEST #3
                     0094
                     0094        ;////////////////////////////////////////////////////////////////////////////////
                     0094  5569          INITIALIZE                                      ;INIT THE CPU
                     0096  5570          EXPUTRAP                                        ;POSSIBLE FAILING MODE
                     0098  5571          LOADDCS         7$,,01,1                        ;MODIFY DCS FOR THIS SUBTEST
                     009F  5572          LOOP            I,1,4                           ;CREATE A TEST LOOP OF 4
                     00A6  5573          LOADREG         WDREG,2$,I,L                    ;LOAD PSL INTO WDREG
                     00AE  5574          ERRLOOP                                         ;ERROR LOOP POINT
                     00B0  5575          FETCH           0                               ;START DCS PROGRAM
                     00B5  5576          BRSTCLK         4                               ;END DCS PROGRAM
                     00B9  5577          CMPREGMSK       CSTRP,8$,I,4$,,W,               ;TEST FOR CORRECT CS ADDRESS
                     00C5  5578          IFERROR         ,<INT>,<UBI>                    ;CS ADDRESS INCORRECT
                     00CC  5579          ENDLOOP         I                               ;END TEST LOOP
                     00CF  5580
                     00CF  5581
                     00CF  5582 ;////////////////////////////////////////////////////////////////////////////////
                     00CF  5583 ;+
                     00CF  5584 ;SUBTEST #4
                     00CF  5585 ;
                     00CF  5586 ;       CURRENT MODE EQUAL 0
                     00CF  5587 ;-
                     00CF  5588          SUBTEST                                         ;SUBTEST #4
                     00D2
                     00D2        ;////////////////////////////////////////////////////////////////////////////////
                     00D2  5589          INITIALIZE                                      ;INIT THE CPU
                     00D4  5590          EXPUTRAP                                        ;POSSIBLE FAILING MODE
                     00D6  5591          LOADDCS         9$,,01,1                        ;MODIFY DCS FOR THIS SUBTEST
                     00DD  5592          LOOP            I,1,4                           ;CREATE A TEST LOOP OF 4
                     00E4  5593          LOADREG         WDREG,2$,I,L                    ;LOAD PSL INTO WDREG
                     00EC  5594          ERRLOOP                                         ;ERROR LOOP POINT
                     00EE  5595          FETCH           0                               ;START DCS PROGRAM
                     00F3  5596          BRSTCLK         4                               ;END DCS PROGRAM
                     00F7  5597          CMPREGMSK       CSTRP,10$,,4$,,W,               ;TEST FOR CORRECT CS ADDRESS
                     0103  5598          IFERROR         ,<INT>,<UBI>                    ;CS ADDRESS INCORRECT
                     010A  5599          ENDLOOP         I                               ;END TEST LOOP
                     010D  5600          SKIP                                            ;GO TO NEXT TEST
                     0114  5601
                     0114  5602
                     0114  5603 BEGIN_TEST_DATA
```

```
                                        0114
                                        0114        ;-----------------------------------------------------------------------
                                        0114  5604
                                        0114  5605 1$:
U 01, 7700,9800,CB70,0301,EA70,1802     0114  5606 ;% 01   PSL(PREV_CURM ISCURM_[3])                ;UCODE SUBTEST #1
                          03C00000      011F  5610 2$:     .LONG   ^X03C00000                       ;PSL TABLE
                          02800000      0123  5611         .LONG   ^X02800000
                          01400000      0127  5612         .LONG   ^X01400000
                          00000000      012B  5613         .LONG   ^X00000000
                              1808      012F  5614 3$:     .WORD   ^X1808                           ;RESULTS SUBTEST #1
                              180A      0131  5615         .WORD   ^X180A
                              180A      0133  5616         .WORD   ^X180A
                              180A      0135  5617         .WORD   ^X180A
                              3FFF      0137  5618 4$:     .WORD   ^X3FFF                           ;MASK FOR CMPREGMSK PSEUDO OP
                                        0139  5619 5$:
U 01, 7700,D800,CB70,0301,6A70,1802     0139  5620 ;% 01   PSL(PREV_CURM ISCURM_[2])                ;UCODE SUBTEST #2
                              1808      0144  5624 6$:     .WORD   ^X1808                           ;RESULTS SUBTEST #2
                              1808      0146  5625         .WORD   ^X1808
                              180A      0148  5626         .WORD   ^X180A
                              180A      014A  5627         .WORD   ^X180A
                                        014C  5628 7$:
U 01, 7700,D800,CB70,0300,EA70,1802     014C  5629 ;% 01   PSL(PREV_CURM ISCURM_[1])                ;UCODE SUBTEST #3
                              1808      0157  5633 8$:     .WORD   ^X1808                           ;RESULTS SUBTEST #3
                              1808      0159  5634         .WORD   ^X1808
                              1808      015B  5635         .WORD   ^X1808
                              180A      015D  5636         .WORD   ^X180A
                                        015F  5637 9$:
U 01, 7700,9800,CB70,0300,6A70,1802     015F  5638 ;% 01   PSL(PREV_CURM ISCURM_[0])                ;UCODE SUBTEST #4
                              1808      016A  5642 10$:    .WORD   ^X1808                           ;RESULTS SUBTEST #4
                                        016C  5643
                                        016C  5644
                                        016C  5645
                                        016C  5646 BEGIN_CPU_DATA
                                        0002  5647
                                        0002  5648 DCS_DATA
                                        0001  5649
U 00, 7700,C800,0364,0300,0470,1801     0001  5650 ;% 00   NOP
U 01, 7700,9800,CB70,0301,EA70,1802     000C  5654 ;% 01   PSL(PREV_CURM ISCURM_[3])                ;SET UP CURRENT MODE
U 02, 7700,1800,CB70,0302,7070,1803     0017  5658 ;% 02   ASTLVL_[4]                               ;INVALIDATE ASTLVL REGISTER
U 03, 5700,C800,0364,7B00,6E90,9808     0022  5662 ;% 03   WB_RDM,REI CHECK?,NEXT/1808              ;PERFORM REI CHECK
U 04, 7300,4800,0364,0300,0470,1805     002D  5666 ;% 04   NOP,STOP.CLOCK                           ;STOP THE CPU CLOCK
                                        0038  5670
                                        0038  5671 END_CPU_DATA
                                        016C  5672
                                        016C  5673
                                        016C  5674 END_TEST_DATA
                                        016C
                                        016C        ;-----------------------------------------------------------------------
                                        016C  5675 ENDTEST
```

```
0015          .SBTTL  "TEST   07F     REI CHECK TEST 3
0015  5677 ;*******************************************************************************
0015  5678 ;++
0015  5679 ;
0015  5680 ; FUNCTIONAL DESCRIPTION:
0015  5681 ;
0015  5682 ;       THIS TEST IS ONE IN A SERIES OF FIVE THAT VERIFIES THE OPERATION
0015  5683 ;       OF REI CHECK.
0015  5684 ;
0015  5685 ;
0015  5686 ; TEST ALGORITHM:
0015  5687 ;
0015  5688 ;       1.  LOAD MICRO CODE FOR SUBTEST
0015  5689 ;       2.  CREATE A TEST LOOP OF 3 (SUBTEST #2)
0015  5690 ;       3.  LOAD RDM WDREG WITH PSL
0015  5691 ;       4.  EXECUTE DCS PROGRAM
0015  5692 ;               a.  INVALIDATE AST REGISTER BY SETTING TO 4
0015  5693 ;               b.  SET 'IS' BIT IN INT TO 1 (SUBTEST #2)
0015  5694 ;               c.  PUT PSL ON WBUS WHILE PERFORMING REI CHECK
0015  5695 ;       5.  TEST CS ADDRESS FOR CORRECT BRANCH
0015  5696 ;       6.  IF NO ERROR GO TO STEP 3 FOR REMAINING PSL'S (SUBTEST #2)
0015  5697 ;
0015  5698 ;
0015  5699 ; TEST PATTERNS:
0015  5700 ;
0015  5701 ;       SUBTEST #1:
0015  5702 ;               WBUS PSL 'IS' BIT = 1 COMPARED TO INT 'IS' BIT = 0
0015  5703 ;
0015  5704 ;       SUBTEST #2: (INT 'IS' BIT = 1)
0015  5705 ;               LOOP 1: WBUS 'IS' BIT = 1 AND CURRENT MODE .NE. 0
0015  5706 ;               LOOP 2: WBUS 'IS' BIT = 1 AND IPL = 0
0015  5707 ;               LOOP 3: WBUS IPL .GT. 0 AND CURRENT MODE .NE. 0
0015  5708 ;
0015  5709 ;
0015  5710 ; LOGIC DESCRIPTION:
0015  5711 ;
0015  5712 ;       THIS TEST CHECKS THAT THE INT GATE ARRAY CAN EXECUTE AN REI CHECK
0015  5713 ;       MICRO ORDER.  SOME OF THE MISCELLANEOUS COMPARISONS THAT COULD NOT
0015  5714 ;       BE PICKED UP IN THE OTHER TESTS ARE DONE HERE. ANY FAILURES SHOULD BE
0015  5715 ;       CONFINED TO THE INT GATE ARRAY.
0015  5716 ;
0015  5717 ;
0015  5718 ; ASSUMPTIONS:
0015  5719 ;
0015  5720 ;       THIS TEST ASSUMES THE WBUS IS OPERATIONAL.
0015  5721 ;
0015  5722 ;
0015  5723 ; ERROR DESCRIPTION:
0015  5724 ;
0015  5725 ;       EXPECTED CS ADDRESS
0015  5726 ;       RECEIVED CS ADDRESS
0015  5727 ;       LOOP COUNT I (SUBTEST #2)
0015  5728 ;
0015  5729 ;--
0015  5730 ;*******************************************************************************
```

```
                        0015  5731 ;////////////////////////////////////////////////////////////////////////////////
                        0015  5732 ;+
                        0015  5733 ;SUBTEST #1
                        0015  5734 ;
                        0015  5735 ;-
                        0015  5736         SUBTEST                                         ;SUBTEST #1
                        0018
                        0018       ;////////////////////////////////////////////////////////////////////////////////
                        0018  5737         INITIALIZE                                      ;INIT THE CPU
                        001A  5738         EXPUTRAP                                        ;POSSIBLE FAILING MODE
                        001C  5739         LOADDCS         1$,,02,1                        ;LOAD UCODE FOR THIS SUBTEST
                        0023  5740         LOADREG         WDREG,2$,,L                     ;LOAD PSL INTO WDREG
                        002B  5741         ERRLOOP                                         ;ERROR LOOP POINT
                        002D  5742         FETCH           0                               ;START DCS PROGRAM
                        0032  5743         BRSTCLK         4                               ;END DCS PROGRAM
                        0036  5744         CMPREGMSK       CSTRP,3$,,4$,,W,                ;TEST FOR CORRECT CS ADDRESS
                        0042  5745         IFERROR         ,<INT>,<UBI>                    ;CS ADDRESS INCORRECT
                        0049  5746
                        0049  5747
                        0049  5748
                        0049  5749 ;////////////////////////////////////////////////////////////////////////////////
                        0049  5750 ;+
                        0049  5751 ;SUBTEST #2
                        0049  5752 ;
                        0049  5753 ;-
                        0049  5754         SUBTEST                                         ;SUBTEST #2
                        004C
                        004C       ;////////////////////////////////////////////////////////////////////////////////
                        004C  5755         INITIALIZE                                      ;INIT THE CPU
                        004E  5756         EXPUTRAP                                        ;POSSIBLE FAILING MODE
                        0050  5757         LOADDCS         5$,,02,1                        ;MODIFY UCODE FOR THIS SUBTEST
                        0057  5758         LOOP            I,1,3                           ;CREATE TEST LOOP OF 3
                        005E  5759         LOADREG         WDREG,6$,I,L                    ;LOAD PSL INTO WDREG
                        0066  5760         ERRLOOP                                         ;ERROR LOOP POINT
                        0068  5761         FETCH           0                               ;START DCS PROGRAM
                        006D  5762         BRSTCLK         4                               ;END DCS PROGRAM
                        0071  5763         CMPREGMSK       CSTRP,3$,,4$,,W,                ;TEST CS ADDRESS
                        007D  5764         IFERROR         ,<INT>,<UBI>                    ;INCORRECT CS ADDRESS
                        0084  5765         ENDLOOP         I                               ;END TEST LOOP
                        0087  5766         SKIP                                            ;GO TO NEXT TEST
                        008E  5767
                        008E  5768 BEGIN_TEST_DATA
                        008E
                        008E       ;-------------------------------------------------------------------------------
                        008E  5769
                        008E  5770 1$:
U 02, 7700,4800,0364,0300,0470,1803  008E  5771 ;% 02    NOP                                 ;UCODE SUBTEST #1
                    04010000  0099  5775 2$:      .LONG    ^X04010000                      ;PSL FOR SUBTEST #1
                        180A  009D  5776 3$:      .WORD    ^X180A                          ;TEST RESULTS
                        3FFF  009F  5777 4$:      .WORD    ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                              00A1  5778 5$:
U 02, 7700,9800,CB70,0302,6A70,1803  00A1  5779 ;% 02    PSL(PREV_CURM ISCURM_[4])           ;UCODE FOR SUBTEST #2
                    05010000  00AC  5783 6$:      .LONG    ^X05010000                      ;PSL TABLE FOR SUBTEST #2
                    04000000  00B0  5784          .LONG    ^X04000000
                    01010000  00B4  5785          .LONG    ^X01010000
```

```
                                00B8  5786
                                00B8  5787
                                00B8  5788 BEGIN_CPU_DATA
                                0002  5789
                                0002  5790 DCS_DATA
                                0001  5791
U 00. 7700.C800.0364.0300.0470.1801  0001  5792 ;% 00   NOP
U 01. 7700.9800.CB70.0302.7070.1802  000C  5796 ;% 01   ASTLVL_[4]                          ;INVALIDATE AST REGISTER
U 02. 7700.4800.0364.0300.0470.1803  0017  5800 ;% 02   NOP
U 03. 5700.C800.0364.7B00.6E90.9808  0022  5804 ;% 03   WB_RDM,REI CHECK?,NEXT/1808        ;PERFORM REI CHECK
U 04. 7300.4800.0364.0300.0470.1805  002D  5808 ;% 04   NOP,STOP.CLOCK                      ;STOP THE CPU CLOCK
                                0038  5812
                                0038  5813 END_CPU_DATA
                                00B8  5814
                                00B8  5815
                                00B8  5816 END_TEST_DATA
                                00B8
                                00B8       ;-----------------------------------------------------------------------
                                00B8  5817 ENDTEST
```

```
0015        .SBTTL  "TEST   080     REI CHECK TEST 4
0015  5819 ;*******************************************************************************
0015  5820 ;++
0015  5821 ;
0015  5822 ; FUNCTIONAL DESCRIPTION:
0015  5823 ;
0015  5824 ;       THIS TEST IS ONE IN A SERIES OF FIVE THAT VERIFIES THE OPERATION
0015  5825 ;       OF REI CHECK.
0015  5826 ;
0015  5827 ;
0015  5828 ; TEST ALGORITHM:
0015  5829 ;
0015  5830 ;       1.  CREATE A TEST LOOP OF 16 TO SELECT PSL'S
0015  5831 ;       2.  LOAD PSL INTO RDM WDREG
0015  5832 ;       3.  EXECUTE DCS PROGRAM
0015  5833 ;               a.  INVALIDATE AST REGISTER BY SETTING TO 4
0015  5834 ;               b.  PUT PSL ON WBUS AND PERFORM REI CHECK
0015  5835 ;       4.  TEST CS ADDRESS FOR PROPER BRANCH
0015  5836 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING PSL'S
0015  5837 ;
0015  5838 ;
0015  5839 ; TEST PATTERNS:
0015  5840 ;
0015  5841 ;       ITERATION       PSL             CS ADDRESS
0015  5842 ;           1           03C00000            1808
0015  5843 ;           2           02C00000            1808
0015  5844 ;           3           01C00000            1808
0015  5845 ;           4           00C00000            1808
0015  5846 ;           5           03800000            180A
0015  5847 ;           6           02800000            1808
0015  5848 ;           7           01800000            1808
0015  5849 ;           8           00800000            1808
0015  5850 ;           9           03400000            180A
0015  5851 ;           10          02400000            180A
0015  5852 ;           11          01400000            1808
0015  5853 ;           12          00400000            1808
0015  5854 ;           13          03000000            180A
0015  5855 ;           14          02000000            180A
0015  5856 ;           15          01000000            180A
0015  5857 ;           16          00000000            1808
0015  5858 ;
0015  5859 ;
0015  5860 ; LOGIC DESCRIPTION:
0015  5861 ;
0015  5862 ;       THIS TEST INSURES THAT THE INT GATE ARRAY CAN PERFORM AN REI CHECK
0015  5863 ;       ON THE COMPARISON BETWEEN WBUS PSL PREVIOUS MODE AND CURRENT MODE.
0015  5864 ;       ANY FAILURES SHOULD BE CONFINED TO THE INT GATE ARRAY.
0015  5865 ;
0015  5866 ;
0015  5867 ; ASSUMPTIONS:
0015  5868 ;
0015  5869 ;       THIS TEST ASSUMES THE WBUS IS OPERATIONAL.
0015  5870 ;
0015  5871 ;
0015  5872 ; ERROR DESCRIPTION:
```

```
                     0015  5873 ;
                     0015  5874 ;       EXPECTED CS ADDRESS
                     0015  5875 ;       RECEIVED CS ADDRESS
                     0015  5876 ;       LOOP COUNT I
                     0015  5877 ;
                     0015  5878 ;--
                     0015  5879 ;*******************************************************************************
                     0015  5880
                     0015  5881
                     0015  5882         INITIALIZE                                  ;INIT THE CPU
                     0017  5883         EXPUTRAP                                    ;POSSIBLE FAILURE MODE
                     0019  5884         LOOP            I,1,16                      ;CREATE A TEST LOOP OF 4
                     0020  5885         LOADREG         WDREG,1$,I,L                ;LOAD PSL INTO WDREG
                     0028  5886         ERRLOOP                                     ;ERROR LOOP POINT
                     002A  5887         FETCH           0                           ;START DCS PROGRAM
                     002F  5888         BRSTCLK         3                           ;END DCS PROGRAM
                     0033  5889         CMPREGMSK       CSTRP,2$,I,3$,,W,           ;TEST CS ADDRESS
                     003F  5890         IFERROR         ,<INT>,<UBI>                ;INCORRECT CS ADDRESS
                     0046  5891         ENDLOOP         I                           ;END TEST LOOP
                     0049  5892         SKIP                                        ;GO TO NEXT TEST
                     0050  5893
                     0050  5894
                     0050  5895 BEGIN_TEST_DATA
                     0050
                     0050       ;-------------------------------------------------------------------------
                     0050  5896
03C00000             0050  5897 1$:      .LONG   ^X03C00000                          ;PSL TABLE
02C00000             0054  5898         .LONG   ^X02C00000
01C00000             0058  5899         .LONG   ^X01C00000
00C00000             005C  5900         .LONG   ^X00C00000
03800000             0060  5901         .LONG   ^X03800000
02800000             0064  5902         .LONG   ^X02800000
01800000             0068  5903         .LONG   ^X01800000
00800000             006C  5904         .LONG   ^X00800000
03400000             0070  5905         .LONG   ^X03400000
02400000             0074  5906         .LONG   ^X02400000
01400000             0078  5907         .LONG   ^X01400000
00400000             007C  5908         .LONG   ^X00400000
03000000             0080  5909         .LONG   ^X03000000
02000000             0084  5910         .LONG   ^X02000000
01000000             0088  5911         .LONG   ^X01000000
00000000             008C  5912         .LONG   ^X00000000
    1808             0090  5913 2$:      .WORD   ^X1808                              ;RESULTS TABLE
    1808             0092  5914         .WORD   ^X1808
    1808             0094  5915         .WORD   ^X1808
    1808             0096  5916         .WORD   ^X1808
    180A             0098  5917         .WORD   ^X180A
    1808             009A  5918         .WORD   ^X1808
    1808             009C  5919         .WORD   ^X1808
    1808             009E  5920         .WORD   ^X1808
    180A             00A0  5921         .WORD   ^X180A
    180A             00A2  5922         .WORD   ^X180A
    1808             00A4  5923         .WORD   ^X1808
    1808             00A6  5924         .WORD   ^X1808
    180A             00A8  5925         .WORD   ^X180A
```

```
                        180A  00AA  5926          .WORD   ^X180A
                        180A  00AC  5927          .WORD   ^X180A
                        1808  00AE  5928          .WORD   ^X1808
                        3FFF  00B0  5929 3$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                              00B2  5930
                              00B2  5931
                              00B2  5932 BEGIN_CPU_DATA
                              0002  5933
                              0002  5934 DCS_DATA
                              0001  5935
U 00, 7700,C800,0364,0300,0470,1801   0001  5936 ;% 00   NOP
U 01, 7700,9800,CB70,0302,7070,1802   000C  5940 ;% 01   ASTLVL_[4]                       ;INVALIDATE AST REGISTER
U 02, 5700,C800,0364,7B00,6E90,9808   0017  5944 ;% 02   WB_RDM,REI CHECK?,NEXT/1808     ;PERFORM REI CHECK
U 03, 7300,C800,0364,0300,0470,1804   0022  5948 ;% 03   NOP,STOP.CLOCK                  ;STOP THE CPU CLOCK
                              002D  5952
                              002D  5953 END_CPU_DATA
                              00B2  5954
                              00B2  5955
                              00B2  5956 END_TEST_DATA
                              00B2
                              00B2        ;-----------------------------------------------------------------------------
                              00B2  5957 ENDTEST
```

```
0015          .SBTTL  "TEST   081     REI CHECK TEST 5
0015  5959 ;*******************************************************************************
0015  5960 ;++
0015  5961 ;
0015  5962 ; FUNCTIONAL DESCRIPTION:
0015  5963 ;
0015  5964 ;       THIS IS ONE OF A SERIES OF FIVE TESTS THAT VERIFIES THE OPERATION
0015  5965 ;       OF REI CHECK.
0015  5966 ;
0015  5967 ;
0015  5968 ; TEST ALGORITHM:
0015  5969 ;
0015  5970 ;       1.  CREATE A TEST LOOP OF 12
0015  5971 ;       2.  LOAD THE RDM WDREG WITH A TEST PSL
0015  5972 ;       3.  MODIFY THE IPL TO BE LOADED INTO THE INT GATE ARRAY
0015  5973 ;       4.  LOAD THE MODIFIED MICRO WORD INTO DCS
0015  5974 ;       5.  EXECUTE DCS PROGRAM
0015  5975 ;               a.  INVALIDATE AST REGISTER BY LOADING WITH 4
0015  5976 ;               b.  LOAD IPL REGISTER IN INT GATE ARRAY
0015  5977 ;               c.  PUT PSL ON WBUS WHILE PERFORMING REI CHECK
0015  5978 ;       6.  TEST CS ADDRESS FOR CORRECT BRANCH
0015  5979 ;       7.  IF NO ERROR GO TO STEP 2 FOR REMAINING PSL'S
0015  5980 ;
0015  5981 ;
0015  5982 ; TEST PATTERNS:
0015  5983 ;
0015  5984 ;       ITERATION       IPL     PSL (WBUS)      CS ADDRESS
0015  5985 ;           1           1E      001F0000        180A
0015  5986 ;           2           1E      001D0000        1808
0015  5987 ;           3           00      00020000        180A
0015  5988 ;           4           03      00030000        1808
0015  5989 ;           5           10      000F0000        1808
0015  5990 ;           6           00      00040000        180A
0015  5991 ;           7           08      00070000        1808
0015  5992 ;           8           04      00030000        1808
0015  5993 ;           9           00      00080000        180A
0015  5994 ;          10           00      00100000        180A
0015  5995 ;          11           00      00000000        1808
0015  5996 ;          12           00      00010000        180A
0015  5997 ;
0015  5998 ;
0015  5999 ; LOGIC DESCRIPTION:
0015  6000 ;
0015  6001 ;       THIS TEST CHECKS THE ABILITY OF THE INT GATE ARRAY TO PERFORM AN REI
0015  6002 ;       CHECK MICRO ORDER.  IN THIS TEST THE IPL CONTAINED IN THE INT ARRAY
0015  6003 ;       IS COMPARED AGAINST THE IPL ON THE WBUS.  ANY FAILURES SHOULD BE
0015  6004 ;       CONFINED TO THE INT ARRAY.
0015  6005 ;
0015  6006 ;
0015  6007 ; ASSUMPTIONS:
0015  6008 ;
0015  6009 ;       THIS TEST ASSUMES THE WBUS IS OPERATIONAL.
0015  6010 ;
0015  6011 ;
0015  6012 ; ERROR DESCRIPTION:
```

```
                        0015  6013 ;
                        0015  6014 ;       EXPECTED CS ADDRESS
                        0015  6015 ;       RECEIVED CS ADDRESS
                        0015  6016 ;       LOOP COUNT I
                        0015  6017 ;
                        0015  6018 ;--
                        0015  6019 ;*******************************************************************************
                        0015  6020
                        0015  6021
                        0015  6022         INITIALIZE                                  ;INIT THE CPU
                        0017  6023         EXPUTRAP                                    ;POSSIBLE FAILURE MODE
                        0019  6024         LOOP            I,1,12                      ;CREATE A TEST LOOP OF 12
                        0020  6025         LOADREG         WDREG,3$,I,L                ;LOAD PSL INTO WDREG
                        0028  6026         LDFIELD         2$,I,B,1$,31,39             ;LOAD LITRL FIELD OF UWORD
                        0039  6027         LOADDCS         1$,,02,1                    ;LOAD UCODE FOR THIS LOOP
                        0040  6028         ERRLOOP                                     ;ERROR LOOP POINT
                        0042  6029         FETCH           0                           ;START DCS PROGRAM
                        0047  6030         BRSTCLK         4                           ;END DCS PROGRAM
                        004B  6031         CMPREGMSK       CSTRP,4$,I,5$,,W,           ;TEST CS ADDRESS
                        0057  6032         IFERROR         ,<INT>,<UBI>                ;WRONG BRANCH ADDRESS
                        005E  6033         ENDLOOP         I                           ;END TEST LOOP
                        0061  6034         SKIP                                        ;GO TO NEXT TEST
                        0068  6035
                        0068  6036
                        0068  6037 BEGIN_TEST_DATA
                        0068
                        0068               ;-------------------------------------------------------------------
                        0068  6038
                        0068  6039 1$:
U 01, 7700,1800,D370,0300,7A70,1802  0068  6040 ;% 01    IPL_[0]                     ;SET UP ACCORDING TO LOOP COUNT
                              1E     0073  6044 2$:      .BYTE   ^X1E                ;TABLE TO LOAD IPL IN INT ARRAY
                              1E     0074  6045          .BYTE   ^X1E
                              00     0075  6046          .BYTE   ^X00
                              03     0076  6047          .BYTE   ^X03
                              10     0077  6048          .BYTE   ^X10
                              00     0078  6049          .BYTE   ^X00
                              08     0079  6050          .BYTE   ^X08
                              04     007A  6051          .BYTE   ^X04
                              00     007B  6052          .BYTE   ^X00
                              00     007C  6053          .BYTE   ^X00
                              00     007D  6054          .BYTE   ^X00
                              00     007E  6055          .BYTE   ^X00
                        001F0000     007F  6056 3$:      .LONG   ^X001F0000          ;TABLE OF PSL'S FOR WBUS
                        001D0000     0083  6057          .LONG   ^X001D0000
                        00020000     0087  6058          .LONG   ^X00020000
                        00030000     008B  6059          .LONG   ^X00030000
                        000F0000     008F  6060          .LONG   ^X000F0000
                        00040000     0093  6061          .LONG   ^X00040000
                        00070000     0097  6062          .LONG   ^X00070000
                        00030000     009B  6063          .LONG   ^X00030000
                        00080000     009F  6064          .LONG   ^X00080000
                        00100000     00A3  6065          .LONG   ^X00100000
                        00000000     00A7  6066          .LONG   ^X00000000
                        00010000     00AB  6067          .LONG   ^X00010000
                            180A     00AF  6068 4$:      .WORD   ^X180A              ;TABLE OF RESULTS
```

```
                        1808  00B1  6069          .WORD   ^X1808
                        180A  00B3  6070          .WORD   ^X180A
                        1808  00B5  6071          .WORD   ^X1808
                        1808  00B7  6072          .WORD   ^X1808
                        180A  00B9  6073          .WORD   ^X180A
                        1808  00BB  6074          .WORD   ^X1808
                        1808  00BD  6075          .WORD   ^X1808
                        180A  00BF  6076          .WORD   ^X180A
                        180A  00C1  6077          .WORD   ^X180A
                        1808  00C3  6078          .WORD   ^X1808
                        180A  00C5  6079          .WORD   ^X180A
                        3FFF  00C7  6080 5$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                              00C9  6081
                              00C9  6082
                              00C9  6083 BEGIN_CPU_DATA
                              0002  6084
                              0002  6085 DCS_DATA
                              0001  6086
U 00. 7700,C800,0364,0300,0470,1801   0001  6087 ;% 00    NOP
U 01. 7700,9800,CB70,0302,7070,1802   000C  6091 ;% 01    ASTLVL_[4]                      ;INVALIDATE AST REGISTER
U 02. 7700,9800,D370,0300,7A70,1803   0017  6095 ;% 02    IPL_[0]                         ;SET UP IPL IN INT GATE ARRAY
U 03. 5700,C800,0364,7B00,6E90,9808   0022  6099 ;% 03    WB_RDM,REI CHECK?,NEXT/1808     ;PERFORM REI CHECK
U 04. 7300,4800,0364,0300,0470,1805   002D  6103 ;% 04    NOP,STOP.CLOCK                  ;STOP THE CPU CLOCK
                              0038  6107
                              0038  6108 END_CPU_DATA
                              00C9  6109
                              00C9  6110
                              00C9  6111 END_TEST_DATA
                              00C9
                              00C9        ;-----------------------------------------------------------------------------
                              00C9  6112 ENDTEST
```

```
                        0019        .SBTTL  "TEST   082     PC Increment on IRD1
                        0019  6114 ;*******************************************************************************
                        0019  6115 ;++
                        0019  6116 ;
                        0019  6117 ; FUNCTIONAL DESCRIPTION
                        0019  6118 ;
                        0019  6119 ;       This test will exaine the PC+2 increment on a BUT/IRD1TST.
                        0019  6120 ;
                        0019  6121 ; TEST ALGORITHM
                        0019  6122 ;
                        0019  6123 ;         1. Initialize the CPU.
                        0019  6124 ;         2. Expect addresses outside of DCS range.
                        0019  6125 ;         3. Load the WDREG with all 0.
                        0019  6126 ;         4. Execute micro-code to:
                        0019  6127 ;                 1. Write the VA with the WDREG.
                        0019  6128 ;                 2. Load a model instruction into the cache.
                        0019  6129 ;                 3. Load the PC with the WDREG.
                        0019  6130 ;                 4. Execute a BUT/IRD1TST.
                        0019  6131 ;                 5. Halt.
                        0019  6132 ;         5. Execute more micro-code to:
                        0019  6133 ;                 6. Read the value of the PC to the RDM WHREG.
                        0019  6134 ;                 7. Halt.
                        0019  6135 ;         6. Compare the WHREG with the expected result of 2.
                        0019  6136 ;         7. If the two are not equal, signal an error.
                        0019  6137 ;         8. End of the test.
                        0019  6138 ;
                        0019  6139 ; TEST DATA
                        0019  6140 ;
                        0019  6141 ;       The only data needed for this test is the initial value of 0 for the
                        0019  6142 ;       VA and the PC.
                        0019  6143 ;
                        0019  6144 ; LOGICAL DESCRIPTION
                        0019  6145 ;
                        0019  6146 ;       On a BUT/IRD1, the PC will always be incremented by two.
                        0019  6147 ;
                        0019  6148 ; ASSUMPTIONS
                        0019  6149 ;
                        0019  6150 ;       This test assumes that the CACHE, XB0, XB1 the IR and the OSR have
                        0019  6151 ;       all been tested.
                        0019  6152 ;
                        0019  6153 ; ERROR DESCRIPTION
                        0019  6154 ;
                        0019  6155 ;       On error this test will type out the following information:
                        0019  6156 ;
                        0019  6157 ;       EXPECTED PC VALUE
                        0019  6158 ;       RECEIVED PC VALUE
                        0019  6159 ;
                        0019  6160 ;--
                        0019  6161 ;*******************************************************************************
                        0019  6162 ;///////////////////////////////////////////////////////////////////////////////
                        0019  6163
                        0019  6164         INITIALIZE                          ; INITIALIZE THE CPU
                        001B  6165         EXPUTRAP                            ; EXPECT ADDRESSES OUTSIDE OF DCS RANGE
                        001D  6166         LOADREG         WDREG,1$,,L         ; LOAD A ZERO INTO THE WDREG
                        0025  6167         ERRLOOP                             ; LOOP ON ERROR FROM HERE
```

```
                                0027  6168          FETCH           0                       ; EXECUTE THE MICRO-CODE FOR AN IRD1
                                002C  6169          BRSTCLK         <^X0B>,INH              ; ...
                                0030  6170          FETCH           <^X0C>                  ; EXECUTE THE MICRO-CODE TO READ THE PC
                                0035  6171          BRSTCLK         <^X0E>,INH              ; ...
                                0039  6172          CMPREG          WHREG,2$,,L             ; DOES THE WHREG = 2?
                                0042  6173          IFERROR         ,,<<MIC>>               ; IF NOT SIGNAL AN ERROR
                                0048  6174          SKIP                                    ; END OF TEST
                                004F  6175
                                004F  6176          BEGIN_TEST_DATA
                                004F
                                004F              ;-------------------------------------------------------------------------
                                004F  6177
                    00000000    004F  6178  1$:     .LONG   0
                    00000002    0053  6179  2$:     .LONG   ^X 00000002
                                0057  6180
                                0057  6181          BEGIN_CPU_DATA
                                0002  6182
                                0002  6183          DCS_DATA
                                0001  6184
U 00, 7700,C800,0364,0300,0470,1801   0001  6185  ;% 00   NOP                             ; LET THE CPU GET UP TO SPEED
U 01, 5700,C800,0364,0300,4A70,1802   000C  6189  ;% 01   VA_RDM                          ; LOAD THE VA WITH ZERO
U 02, 7700,F800,57D7,D7EF,0470,1803   0017  6193  ;% 02   LONLIT_[50505021]              ; LOAD A MODEL INSTRUCTION INTO LONLIT
U 03, 7701,8800,5BE4,02D4,5590,1804   0022  6197  ;% 03   WRITE.LONG R[LONLIT]           ; WRITE LONLIT TO CACHE
U 04, 7700,4800,5BE4,03D4,5470,1805   002D  6201  ;% 04   WDR_UNROT(R[LONLIT])           ; ...
U 05, 7700,C800,0364,0300,4470,1806   0038  6205  ;% 05   VA_VA+4                        ; INCREMENT THE VA
U 06, 7700,7800,57D7,D7D7,8470,1807   0043  6209  ;% 06   LONLIT_[50505050]             ; WRITE THE REST OF THE INSTRUCTION TO
U 07, 7701,8800,5BE4,02D4,5590,1808   004E  6213  ;% 07   WRITE.LONG R[LONLIT]           ; CACHE
U 08, 7700,4800,5BE4,03D4,5470,1809   0059  6217  ;% 08   WDR_UNROT(R[LONLIT])           ; ...
U 09, 5700,C800,0364,0300,4870,180A   0064  6221  ;% 09   PC_RDM                        ; START INSTRUCTION PREFETCH
U 0A, 7700,C800,0364,1700,0470,180B   006F  6225  ;% 0A   BUT/IRD1TST                   ; EXECUTE THE IRD1TST
U 0B, 7300,4800,0364,0300,0470,180C   007A  6229  ;% 0B   STOP.CLOCK                    ; HALT
U 0C, 7700,C800,0364,0300,0470,180D   0085  6233  ;% 0C   NOP                           ; LET THE CPU GET UP TO SPEED
U 0D, 6700,C81A,0024,03D8,0470,180E   0090  6237  ;% 0D   RDM_PC                       ; READ BACK THE PC
U 0E, 7300,4800,0364,0300,0470,180F   009B  6241  ;% 0E   STOP.CLOCK                    ; HALT
                                00A6  6245
                                00A6  6246  END_CPU_DATA
                                0057  6247
                                0057  6248  END_TEST_DATA
                                0057
                                0057        ;-------------------------------------------------------------------------
                                0057  6249
                                0057  6250  ENDTEST
```

```
0021          .SBTTL  "TEST   083     NATIVE MODE DSIZE ROM TEST 1
0021   6252 ;*******************************************************************************
0021   6253 ;++
0021   6254 ;
0021   6255 ; FUNCTIONAL DESCRIPTION:
0021   6256 ;
0021   6257 ;       THIS TEST USE THE HALT INSTRUCTION TO TEST THE DSIZE AND ISIZE
0021   6258 ;       MULTIPLEXERS ON DPM.
0021   6259 ;
0021   6260 ;
0021   6261 ; TEST ALGORITHM:
0021   6262 ;
0021   6263 ;       1.  EXECUTE DCS PROGRAM
0021   6264 ;               a.  DISABLE ^P INTERRUPTS
0021   6265 ;               b.  SET IPL TO 1F
0021   6266 ;               c.  LOAD PC WITH ZERO TO FORCE PREFETCH
0021   6267 ;               d.  PERFORM IRD1TEST TO LOAD THE IR REGISTER WITH HALT INST.
0021   6268 ;               e.  SET ISIZE AND DSIZE TO INST. DEPENDENT AND STROBE VBUS
0021   6269 ;       2.  TEST FOR CORRECT ISIZE AND DSIZE
0021   6270 ;       3.  IF NO ERROR CREATE A TEST LOOP OF 5
0021   6271 ;       4.  EXECUTE DCS PROGRAM
0021   6272 ;               b.  INCREMENT THE IRD COUNTER BY ONE
0021   6273 ;               c.  SET ISIZE AND DSIZE TO INST. DEPENDENT AND STROBE VBUS
0021   6274 ;                       ALSO BRANCH THE CS ADDRESS ON DSIZE
0021   6275 ;       5.  TEST FOR CORRECT ISIZE AND DSIZE
0021   6276 ;       6.  IF NO ERROR TEST THE CS ADDRESS FOR THE CORRECT BRANCH
0021   6277 ;       7.  IF NO ERROR GO TO STEP 4 FOR REMAINING LOOP COUNTS
0021   6278 ;
0021   6279 ;
0021   6280 ; TEST PATTERNS:
0021   6281 ;
0021   6282 ;       HALT INSTRUCTION
0021   6283 ;
0021   6284 ;
0021   6285 ; LOGIC DESCRIPTION:
0021   6286 ;
0021   6287 ;       THIS TEST USES THE HALT INSTRUCTION TO TEST THE DSIZE AND ISIZE
0021   6288 ;       MULTIPLEXERS.  THE DSIZE ROM HAS THE HALT INSTRUCTION BLASTED
0021   6289 ;       WITH ALL THE DATA TYPES POSSIBLE.  SO BY INCREMENTING THE IRD
0021   6290 ;       COUNTER YOU CAN DRIVE THE DSIZE AND ISIZE SIGNALS TO ALL
0021   6291 ;       THE POSSIBLE STATES (BYTE, WORD, LONG, QUAD).  ONE THING TO CHECK,
0021   6292 ;       IF YOU HAVE FAILURES, IS THAT THE IRD COUNTER IN THE SAC GATE ARRAY
0021   6293 ;       INCREMENTS PROPERLY.
0021   6294 ;       IF ERRORS OCCUR IN THE CS BRANCH ADDRESS THEN YOU HAVE A PROBLEM IN
0021   6295 ;       THE DISCRETE LOGIC ON DPM THAT GATES THE DSIZE ONTO CS ADDRESS BITS
0021   6296 ;       <1:0>.
0021   6297 ;
0021   6298 ;
0021   6299 ; ASSUMPTIONS:
0021   6300 ;
0021   6301 ;       THIS TEST ASSUMES THE PREFETCH LOGIC AND IR REGISTER ARE OPERATIONAL.
0021   6302 ;
0021   6303 ;
0021   6304 ; ERROR DESCRIPTION:
0021   6305 ;
```

```
                        6306 ;       FIRST AND SECOND IFERROR:
                        6307 ;               EXPECTED VBUS DATA (DSIZE/ISIZE)
                        6308 ;               RECEIVED VBUS DATA
                        6309 ;               LOOP COUNT I (SECOND IFERROR ONLY)
                        6310 ;                       (BITS <7:0> = VBUS BIT POSITION)
                        6311 ;                       (BITS <8> = BIT STATE)
                        6312 ;
                        6313 ;       THIRD IFERROR:
                        6314 ;               EXPECTED CS ADDRESS
                        6315 ;               RECEIVED CS ADDRESS
                        6316 ;               LOOP COUNT I
                        6317 ;
                        6318 ;--
                        6319 ;******************************************************************************
                        6320
                        6321
0021                    6322         INITIALIZE                                 ;INIT THE CPU
0023                    6323         ERRLOOP                                    ;ERROR LOOP POINT
0025                    6324         FETCH           0                          ;START DCS PROGRAM
002A                    6325         BRSTCLK         7,INH                      ;END DCS PROGRAM
002E                    6326         CMPVBUS         1$                         ;TEST DSIZE & ISIZE
0033                    6327         IFERROR                                    ;BITS NOT CORRECT
0038                    6328         LOOP            I,1,5                      ;CREATE LOOP OF 5
003F                    6329         FETCH           8                          ;START DCS PROGRAM
0044                    6330         BRSTCLK         <^X0B>,INH                 ;END DCS PROGRAM
0048                    6331         CMPVBUS         2$,I                       ;TEST DSIZE & ISIZE
004D                    6332         IFERROR         ,<SAC>,                    ;BITS NOT CORRECT
0053                    6333         CMPREGMSK       CSTRP,3$,I,4$,,W,          ;TEST CS ADDRESS
005F                    6334         IFERROR                                    ;INCORRECT BRANCH ADDRESS
0064                    6335         ENDLOOP         I                          ;END TEST LOOP
0067                    6336         SKIP
006E                    6337
006E                    6338
006E                    6339 BEGIN_TEST_DATA
006E
006E                            ;-------------------------------------------------------------------------
006E                    6340
04  006E                6341 1$:     .BYTE   ^X4                                ;RESULTS FOR DSIZE & ISIZE
    006F                6342         VBUSDATA        <^X20>,0
    0070                6343         VBUSDATA        <^X21>,1
    0071                6344         VBUSDATA        <^X24>,0
    0072                6345         VBUSDATA        <^X25>,0
04  0073                6346 2$:     .BYTE   ^X4
    0074                6347         VBUSDATA        <^X20>,1
    0075                6348         VBUSDATA        <^X21>,0
    0076                6349         VBUSDATA        <^X24>,1
    0077                6350         VBUSDATA        <^X25>,0
04  0078                6351         .BYTE   ^X4
    0079                6352         VBUSDATA        <^X20>,0
    007A                6353         VBUSDATA        <^X21>,0
    007B                6354         VBUSDATA        <^X24>,0
    007C                6355         VBUSDATA        <^X25>,1
04  007D                6356         .BYTE   ^X4
    007E                6357         VBUSDATA        <^X20>,0
    007F                6358         VBUSDATA        <^X21>,0
```

```
                                0080  6359          VBUSDATA        <^X24>,1
                                0081  6360          VBUSDATA        <^X25>,1
                          04    0082  6361          .BYTE   ^X4
                                0083  6362          VBUSDATA        <^X20>,1
                                0084  6363          VBUSDATA        <^X21>,0
                                0085  6364          VBUSDATA        <^X24>,1
                                0086  6365          VBUSDATA        <^X25>,0
                          04    0087  6366          .BYTE   ^X4
                                0088  6367          VBUSDATA        <^X20>,0
                                0089  6368          VBUSDATA        <^X21>,0
                                008A  6369          VBUSDATA        <^X24>,0
                                008B  6370          VBUSDATA        <^X25>,1
                        1801    008C  6371 3$:      .WORD   ^X1801                  ;CS BRANCH ADDRESSES
                        1802    008E  6372          .WORD   ^X1802
                        1803    0090  6373          .WORD   ^X1803
                        1801    0092  6374          .WORD   ^X1801
                        1802    0094  6375          .WORD   ^X1802
                        3FFF    0096  6376 4$:      .WORD   ^X3FFF                  ;MASK FOR CMPREGMSK PSEUDO OP
                                0098  6377
                                0098  6378 BEGIN_CPU_DATA
                                0002  6379
                                0002  6380 DCS_DATA
                                0001  6381
U 00, 7700,C800,0364,0300,0470,1801     0001  6382 ;% 00   NOP
U 01, 7700,4800,5BE4,0300,2470,1802     000C  6386 ;% 01   WCTRL/LOADCRAR,WB_R[TEMP0]      ;DISABLE ^P INTERRUPT
U 02, 7700,0800,5BE4,0304,2C70,1803     0017  6390 ;% 02   WCTRL/CONWRITE,WB_R[TEMP1]     ;...
U 03, 7700,9800,D370,030F,FA70,1804     0022  6394 ;% 03   IPL_[1F]                      ;SET IPL TO 1F
U 04, 7700,C800,5BE4,03D8,4870,1805     002D  6398 ;% 04   PC_R[ZERO]                    ;CAUSE PREFETCH TO HAPPEN
U 05, 7700,4800,0364,1700,0470,1806     0038  6402 ;% 05   IRD1TEST                      ;LOAD THE IR REGISTER
U 06, 7601,4800,0364,0302,0470,1807     0043  6406 ;% 06   ISIZE[IDEP],STROBE.V.BUS     ;SET DSIZE & ISIZE TO IDEP
U 07, 7300,C800,0364,0300,0470,1808     004E  6410 ;% 07   NOP,STOP.CLOCK               ;STOP THE CPU CLOCK
U 08, 7700,4800,0364,0300,0470,1809     0059  6414 ;% 08   NOP
U 09, 7700,4800,0364,1B00,0470,180A     0064  6418 ;% 09   LOD INC BRA?                 ;INCREMENT THE IRD COUNTER
U 0A, 7601,8800,0364,C702,0470,1800     006F  6422 ;% 0A   ISIZE[IDEP],DSIZE?,NEXT/1800,STROBE.V.BUS;SET DSIZE & ISIZE TO IDEP
U 0B, 7300,4800,0364,0300,0470,180C     007A  6426 ;% 0B   NOP,STOP.CLOCK               ;STOP THE CPU CLOCK
                                0085  6430
                                0085  6431 RTEMP_DATA
                                0001  6432
                    00C00000    0001  6433          .LONG   ^X00C00000              ;DISABLE ^P INTERRUPT
                    00400000    0005  6434          .LONG   ^X00400000              ;...
                                0009  6435
                                0009  6436 CACHE_DATA
                                0001  6437
                    00000000    0001  6438          .LONG   ^X00000000              ;HALT INSTRUCTION FOR IR
                    00000000    0005  6439          .LONG   ^X00000000
                                0009  6440
                                0009  6441 END_CPU_DATA
                                0098  6442
                                0098  6443
                                0098  6444 END_TEST_DATA
                                0098
                                0098          ;-----------------------------------------------------------------------
                                0098  6445 ENDTEST
```

```
                        0021        .SBTTL  "TEST   084     NATIVE MODE DSIZE ROM TEST 2
                        0021  6447 ;*******************************************************************************
                        0021  6448 ;++
                        0021  6449 ;
                        0021  6450 ; FUNCTIONAL DESCRIPTION:
                        0021  6451 ;
                        0021  6452 ;       THIS IS ONE IN A SERIES OF 14 TESTS THAT CHECK THE CONTENTS OF THE
                        0021  6453 ;       DSIZE ROM.
                        0021  6454 ;
                        0021  6455 ;
                        0021  6456 ; TEST ALGORITHM:
                        0021  6457 ;
                        0021  6458 ;       1.  CREATE AN INDEX (K) TO INDEX RESULTS TABLE
                        0021  6459 ;       2.  CREATE LOOP (I) OF 25
                        0021  6460 ;       3.  SAVE LOOP COUNT IN 1$
                        0021  6461 ;       4.  MOVE LOOP COUNT TO WDREG
                        0021  6462 ;       5.  EXECUTE DCS PROGRAM
                        0021  6463 ;               a.  DISABLE ^P INTERRUPTS
                        0021  6464 ;               b.  SET IPL TO 1F
                        0021  6465 ;               c.  MOVE LOOP COUNT TO RTEMP2
                        0021  6466 ;               d.  ROTATE LOOP COUNT LEFT 2 TO FORM CACHE ADDRESS
                        0021  6467 ;               e.  LOAD THIS ADDRESS INTO PC TO CAUSE PREFETCH
                        0021  6468 ;               f.  PERFORM IRD1TEST TO CAUSE IR TO LOAD INSTRUCTION
                        0021  6469 ;               g.  SET DTYPE TO INSTURCTION DEPENDENT AND STROBE THE VBUS
                        0021  6470 ;       6.  TEST VBUS FOR CORRECT DSIZE
                        0021  6471 ;       7.  IF NO ERROR CREATE LOOP (J) OF 5
                        0021  6472 ;       8.  EXECUTE DCS PROGRAM
                        0021  6473 ;               a.  INCREMENT THE IRD COUNTER
                        0021  6474 ;               b.  SET DTYPE TO INSTRUCTION DEPENDENT AND STROBE THE VBUS
                        0021  6475 ;       9.  TEST VBUS FOR CORRECT DSIZE
                        0021  6476 ;      10.  INCREMENT INDEX K BY ONE
                        0021  6477 ;      11.  IF NO ERROR GO TO STEP 8 UNTIL LOOP J IS EXHAUSTED
                        0021  6478 ;      12.  GO TO STEP 3 UNTIL LOOP I IS EXHAUSTED
                        0021  6479 ;
                        0021  6480 ;
                        0021  6481 ; TEST PATTERNS:
                        0021  6482 ;
                        0021  6483 ;       LOOP COUNT I                INSTRUCTION
                        0021  6484 ;            1                         NOP
                        0021  6485 ;            2                         REI
                        0021  6486 ;            3                         BPT
                        0021  6487 ;            4                         RET
                        0021  6488 ;            5                         RSB
                        0021  6489 ;            6                         LDPCTX
                        0021  6490 ;            7                         SVPCTX
                        0021  6491 ;            8                         CVTPS
                        0021  6492 ;            9                         CVTSP
                        0021  6493 ;            A                         INDEX
                        0021  6494 ;            B                         CRC
                        0021  6495 ;            C                         PROBER
                        0021  6496 ;            D                         PROBEW
                        0021  6497 ;            E                         INSQUE
                        0021  6498 ;            F                         REMQUE
                        0021  6499 ;           10                         BSBB
                        0021  6500 ;           11                         BRB
```

```
                        0021  6501 ;          12                      BNEQ
                        0021  6502 ;          13                      BEQL
                        0021  6503 ;          14                      BGTR
                        0021  6504 ;          15                      BLEQ
                        0021  6505 ;          16                      JSB
                        0021  6506 ;          17                      JMP
                        0021  6507 ;          18                      BGEQ
                        0021  6508 ;          19                      BLSS
                        0021  6509 ;
                        0021  6510 ;
                        0021  6511 ; LOGIC DESCRIPTION:
                        0021  6512 ;
                        0021  6513 ;       THIS TEST ALONG WITH THE OTHER 13 TESTS IN THIS SERIES CHECK ALL THE
                        0021  6514 ;       LOCATIONS BLASTED INTO THE DSIZE ROM.  ANY FALIURES SHOULD BE CONFINED
                        0021  6515 ;       TO THE ROM.
                        0021  6516 ;
                        0021  6517 ;
                        0021  6518 ; ASSUMPTIONS:
                        0021  6519 ;
                        0021  6520 ;       THIS TEST ASSUMES THE PREFETCH LOGIC AND INSTRUCTION REGISTER ARE
                        0021  6521 ;       OPERATIONAL.
                        0021  6522 ;
                        0021  6523 ;
                        0021  6524 ; ERROR DESCRIPTION:
                        0021  6525 ;
                        0021  6526 ;       EXPECTED VBUS DATA (DSIZE)
                        0021  6527 ;       RECEIVED VBUS DATA
                        0021  6528 ;       LOOP COUNT I
                        0021  6529 ;       LOOP COUNT J
                        0021  6530 ;       LOOP COUNT K
                        0021  6531 ;              (BITS <7:0> = VBUS BIT POSITION)
                        0021  6532 ;              (BITS <8> = BIT STATE)
                        0021  6533 ;
                        0021  6534 ;--
                        0021  6535 ;*******************************************************************************
                        0021  6536
                        0021  6537
                        0021  6538          INITIALIZE                          ;INIT THE CPU
                        0023  6539          LDINDEX         K,1                 ;INIT INDEX K TO 1
                        002A  6540          LOOP            I,1,25              ;CREATE A TEST LOOP OF 50
                        0031  6541          SAVEINDEX       I,1$                ;SAVE LOOP COUNT IN 1$
                        0036  6542          LOADREG         WDREG,1$,,L         ;LOAD LOOP COUNT INTO WDREG
                        003E  6543          ERRLOOP                             ;ERROR LOOP POINT
                        0040  6544          FETCH           0                   ;START DCS PROGRAM
                        0045  6545          BRSTCLK         <^X0A>,INH          ;END DCS PROGRAM
                        0049  6546          CMPVBUS         2$,I                ;TEST DSIZE
                        004E  6547          IFERROR                             ;ROM FAILURE
                        0053  6548          LOOP            J,1,5               ;CREATE TEST LOOP OF 5
                        005A  6549          FETCH           <^X0B>              ;START DCS PROGRAM
                        005F  6550          BRSTCLK         <^X0E>,INH          ;END DCS PROGRAM
                        0063  6551          CMPVBUS         3$,K                ;TEST DSIZE
                        0068  6552          IFERROR                             ;ROM FAILURE
                        006D  6553          INCINDEX        K                   ;INCREMENT INDEX K BY ONE
                        0070  6554          ENDLOOP         J                   ;END TEST LOOP J
                        0073  6555          ENDLOOP         I                   ;END TEST LOOP I
```

```
                    0076  6556          SKIP                                ;GO TO NEXT TEST
                    007D  6557
                    007D  6558
                    007D  6559 BEGIN_TEST_DATA
                    007D
                    007D          ;-----------------------------------------------------------------------
                    007D  6560
          00000000  007D  6561 1$:      .LONG   ^X00000000                  ;LOOP COUNT STORAGE
                 02 0081  6562 2$:      .BYTE   ^X2                         ;RESULTS TABLE (IRD 0)
                    0082  6563          VBUSDATA        <^X24>,0            ;NOP
                    0083  6564          VBUSDATA        <^X25>,0
                 02 0084  6565          .BYTE   ^X2                         ;REI
                    0085  6566          VBUSDATA        <^X24>,0
                    0086  6567          VBUSDATA        <^X25>,0
                 02 0087  6568          .BYTE   ^X2                         ;BPT
                    0088  6569          VBUSDATA        <^X24>,0
                    0089  6570          VBUSDATA        <^X25>,0
                 02 008A  6571          .BYTE   ^X2                         ;RET
                    008B  6572          VBUSDATA        <^X24>,0
                    008C  6573          VBUSDATA        <^X25>,0
                 02 008D  6574          .BYTE   ^X2                         ;RSB
                    008E  6575          VBUSDATA        <^X24>,0
                    008F  6576          VBUSDATA        <^X25>,1
                 02 0090  6577          .BYTE   ^X2                         ;LDPCTX
                    0091  6578          VBUSDATA        <^X24>,0
                    0092  6579          VBUSDATA        <^X25>,0
                 02 0093  6580          .BYTE   ^X2                         ;SVPCTX
                    0094  6581          VBUSDATA        <^X24>,0
                    0095  6582          VBUSDATA        <^X25>,0
                 02 0096  6583          .BYTE   ^X2                         ;CVTPS
                    0097  6584          VBUSDATA        <^X24>,1
                    0098  6585          VBUSDATA        <^X25>,0
                 02 0099  6586          .BYTE   ^X2                         ;CVTSP
                    009A  6587          VBUSDATA        <^X24>,1
                    009B  6588          VBUSDATA        <^X25>,0
                 02 009C  6589          .BYTE   ^X2                         ;INDEX
                    009D  6590          VBUSDATA        <^X24>,0
                    009E  6591          VBUSDATA        <^X25>,1
                 02 009F  6592          .BYTE   ^X2                         ;CRC
                    00A0  6593          VBUSDATA        <^X24>,0
                    00A1  6594          VBUSDATA        <^X25>,0
                 02 00A2  6595          .BYTE   ^X2                         ;PROBER
                    00A3  6596          VBUSDATA        <^X24>,0
                    00A4  6597          VBUSDATA        <^X25>,0
                 02 00A5  6598          .BYTE   ^X2                         ;PROBEW
                    00A6  6599          VBUSDATA        <^X24>,0
                    00A7  6600          VBUSDATA        <^X25>,0
                 02 00A8  6601          .BYTE   ^X2                         ;INSQUE
                    00A9  6602          VBUSDATA        <^X24>,0
                    00AA  6603          VBUSDATA        <^X25>,0
                 02 00AB  6604          .BYTE   ^X2                         ;REMQUE
                    00AC  6605          VBUSDATA        <^X24>,0
                    00AD  6606          VBUSDATA        <^X25>,0
                 02 00AE  6607          .BYTE   ^X2                         ;BSBB
                    00AF  6608          VBUSDATA        <^X24>,0
```

```
                    00B0  6609          VBUSDATA        <^X25>,0
                02  00B1  6610          .BYTE   ^X2                             ;BRB
                    00B2  6611          VBUSDATA        <^X24>,0
                    00B3  6612          VBUSDATA        <^X25>,0
                02  00B4  6613          .BYTE   ^X2                             ;BNEQ
                    00B5  6614          VBUSDATA        <^X24>,0
                    00B6  6615          VBUSDATA        <^X25>,0
                02  00B7  6616          .BYTE   ^X2                             ;BEQL
                    00B8  6617          VBUSDATA        <^X24>,0
                    00B9  6618          VBUSDATA        <^X25>,0
                02  00BA  6619          .BYTE   ^X2                             ;BGTR
                    00BB  6620          VBUSDATA        <^X24>,0
                    00BC  6621          VBUSDATA        <^X25>,0
                02  00BD  6622          .BYTE   ^X2                             ;BLEQ
                    00BE  6623          VBUSDATA        <^X24>,0
                    00BF  6624          VBUSDATA        <^X25>,0
                02  00C0  6625          .BYTE   ^X2                             ;JSB
                    00C1  6626          VBUSDATA        <^X24>,0
                    00C2  6627          VBUSDATA        <^X25>,0
                02  00C3  6628          .BYTE   ^X2                             ;JMP
                    00C4  6629          VBUSDATA        <^X24>,0
                    00C5  6630          VBUSDATA        <^X25>,0
                02  00C6  6631          .BYTE   ^X2                             ;BGEQ
                    00C7  6632          VBUSDATA        <^X24>,0
                    00C8  6633          VBUSDATA        <^X25>,0
                02  00C9  6634          .BYTE   ^X2                             ;BLSS
                    00CA  6635          VBUSDATA        <^X24>,0
                    00CB  6636          VBUSDATA        <^X25>,0
                    00CC  6637
                02  00CC  6638 3$:      .BYTE   ^X2                             ;RESULTS TABLE (IRD 1 TO 5)
                    00CD  6639          VBUSDATA        <^X24>,0                ;NOP
                    00CE  6640          VBUSDATA        <^X25>,0
                02  00CF  6641          .BYTE   ^X2
                    00D0  6642          VBUSDATA        <^X24>,0
                    00D1  6643          VBUSDATA        <^X25>,0
                02  00D2  6644          .BYTE   ^X2
                    00D3  6645          VBUSDATA        <^X24>,0
                    00D4  6646          VBUSDATA        <^X25>,0
                02  00D5  6647          .BYTE   ^X2
                    00D6  6648          VBUSDATA        <^X24>,0
                    00D7  6649          VBUSDATA        <^X25>,0
                02  00D8  6650          .BYTE   ^X2
                    00D9  6651          VBUSDATA        <^X24>,0
                    00DA  6652          VBUSDATA        <^X25>,0
                02  00DB  6653          .BYTE   ^X2                             ;REI
                    00DC  6654          VBUSDATA        <^X24>,0
                    00DD  6655          VBUSDATA        <^X25>,0
                02  00DE  6656          .BYTE   ^X2
                    00DF  6657          VBUSDATA        <^X24>,0
                    00E0  6658          VBUSDATA        <^X25>,0
                02  00E1  6659          .BYTE   ^X2
                    00E2  6660          VBUSDATA        <^X24>,0
                    00E3  6661          VBUSDATA        <^X25>,0
                02  00E4  6662          .BYTE   ^X2
                    00E5  6663          VBUSDATA        <^X24>,0
```

```
                    00E6  6664          VBUSDATA        <^X25>,0
                02  00E7  6665          .BYTE   ^X2
                    00E8  6666          VBUSDATA        <^X24>,0
                    00E9  6667          VBUSDATA        <^X25>,0
                02  00EA  6668          .BYTE   ^X2                             ;BPT
                    00EB  6669          VBUSDATA        <^X24>,0
                    00EC  6670          VBUSDATA        <^X25>,0
                02  00ED  6671          .BYTE   ^X2
                    00EE  6672          VBUSDATA        <^X24>,0
                    00EF  6673          VBUSDATA        <^X25>,0
                02  00F0  6674          .BYTE   ^X2
                    00F1  6675          VBUSDATA        <^X24>,0
                    00F2  6676          VBUSDATA        <^X25>,0
                02  00F3  6677          .BYTE   ^X2
                    00F4  6678          VBUSDATA        <^X24>,0
                    00F5  6679          VBUSDATA        <^X25>,0
                02  00F6  6680          .BYTE   ^X2
                    00F7  6681          VBUSDATA        <^X24>,0
                    00F8  6682          VBUSDATA        <^X25>,0
                02  00F9  6683          .BYTE   ^X2                             ;RET
                    00FA  6684          VBUSDATA        <^X24>,0
                    00FB  6685          VBUSDATA        <^X25>,0
                02  00FC  6686          .BYTE   ^X2
                    00FD  6687          VBUSDATA        <^X24>,0
                    00FE  6688          VBUSDATA        <^X25>,0
                02  00FF  6689          .BYTE   ^X2
                    0100  6690          VBUSDATA        <^X24>,0
                    0101  6691          VBUSDATA        <^X25>,0
                02  0102  6692          .BYTE   ^X2
                    0103  6693          VBUSDATA        <^X24>,0
                    0104  6694          VBUSDATA        <^X25>,0
                02  0105  6695          .BYTE   ^X2
                    0106  6696          VBUSDATA        <^X24>,0
                    0107  6697          VBUSDATA        <^X25>,0
                02  0108  6698          .BYTE   ^X2                             ;RSB
                    0109  6699          VBUSDATA        <^X24>,0
                    010A  6700          VBUSDATA        <^X25>,0
                02  010B  6701          .BYTE   ^X2
                    010C  6702          VBUSDATA        <^X24>,0
                    010D  6703          VBUSDATA        <^X25>,0
                02  010E  6704          .BYTE   ^X2
                    010F  6705          VBUSDATA        <^X24>,0
                    0110  6706          VBUSDATA        <^X25>,0
                02  0111  6707          .BYTE   ^X2
                    0112  6708          VBUSDATA        <^X24>,0
                    0113  6709          VBUSDATA        <^X25>,0
                02  0114  6710          .BYTE   ^X2
                    0115  6711          VBUSDATA        <^X24>,0
                    0116  6712          VBUSDATA        <^X25>,0
                02  0117  6713          .BYTE   ^X2                             ;LDPCTX
                    0118  6714          VBUSDATA        <^X24>,0
                    0119  6715          VBUSDATA        <^X25>,0
                02  011A  6716          .BYTE   ^X2
                    011B  6717          VBUSDATA        <^X24>,0
                    011C  6718          VBUSDATA        <^X25>,0
```

```
                02  011D  6719          .BYTE   ^X2
                    011E  6720          VBUSDATA        <^X24>,0
                    011F  6721          VBUSDATA        <^X25>,0
                02  0120  6722          .BYTE   ^X2
                    0121  6723          VBUSDATA        <^X24>,0
                    0122  6724          VBUSDATA        <^X25>,0
                02  0123  6725          .BYTE   ^X2
                    0124  6726          VBUSDATA        <^X24>,0
                    0125  6727          VBUSDATA        <^X25>,0
                02  0126  6728          .BYTE   ^X2                             ;SVPCTX
                    0127  6729          VBUSDATA        <^X24>,0
                    0128  6730          VBUSDATA        <^X25>,0
                02  0129  6731          .BYTE   ^X2
                    012A  6732          VBUSDATA        <^X24>,0
                    012B  6733          VBUSDATA        <^X25>,0
                02  012C  6734          .BYTE   ^X2
                    012D  6735          VBUSDATA        <^X24>,0
                    012E  6736          VBUSDATA        <^X25>,0
                02  012F  6737          .BYTE   ^X2
                    0130  6738          VBUSDATA        <^X24>,0
                    0131  6739          VBUSDATA        <^X25>,0
                02  0132  6740          .BYTE   ^X2
                    0133  6741          VBUSDATA        <^X24>,0
                    0134  6742          VBUSDATA        <^X25>,0
                02  0135  6743          .BYTE   ^X2                             ;CVTPS
                    0136  6744          VBUSDATA        <^X24>,0
                    0137  6745          VBUSDATA        <^X25>,0
                02  0138  6746          .BYTE   ^X2
                    0139  6747          VBUSDATA        <^X24>,1
                    013A  6748          VBUSDATA        <^X25>,0
                02  013B  6749          .BYTE   ^X2
                    013C  6750          VBUSDATA        <^X24>,0
                    013D  6751          VBUSDATA        <^X25>,0
                02  013E  6752          .BYTE   ^X2
                    013F  6753          VBUSDATA        <^X24>,0
                    0140  6754          VBUSDATA        <^X25>,0
                02  0141  6755          .BYTE   ^X2
                    0142  6756          VBUSDATA        <^X24>,0
                    0143  6757          VBUSDATA        <^X25>,0
                02  0144  6758          .BYTE   ^X2                             ;CVTSP
                    0145  6759          VBUSDATA        <^X24>,0
                    0146  6760          VBUSDATA        <^X25>,0
                02  0147  6761          .BYTE   ^X2
                    0148  6762          VBUSDATA        <^X24>,1
                    0149  6763          VBUSDATA        <^X25>,0
                02  014A  6764          .BYTE   ^X2
                    014B  6765          VBUSDATA        <^X24>,0
                    014C  6766          VBUSDATA        <^X25>,0
                02  014D  6767          .BYTE   ^X2
                    014E  6768          VBUSDATA        <^X24>,0
                    014F  6769          VBUSDATA        <^X25>,0
                02  0150  6770          .BYTE   ^X2
                    0151  6771          VBUSDATA        <^X24>,0
                    0152  6772          VBUSDATA        <^X25>,0
                02  0153  6773          .BYTE   ^X2                             ;INDEX
```

```
                     0154  6774          VBUSDATA        <^X24>,0
                     0155  6775          VBUSDATA        <^X25>,1
                  02 0156  6776          .BYTE   ^X2
                     0157  6777          VBUSDATA        <^X24>,0
                     0158  6778          VBUSDATA        <^X25>,1
                  02 0159  6779          .BYTE   ^X2
                     015A  6780          VBUSDATA        <^X24>,0
                     015B  6781          VBUSDATA        <^X25>,1
                  02 015C  6782          .BYTE   ^X2
                     015D  6783          VBUSDATA        <^X24>,0
                     015E  6784          VBUSDATA        <^X25>,1
                  02 015F  6785          .BYTE   ^X2
                     0160  6786          VBUSDATA        <^X24>,0
                     0161  6787          VBUSDATA        <^X25>,1
                  02 0162  6788          .BYTE   ^X2                             ;CRC
                     0163  6789          VBUSDATA        <^X24>,0
                     0164  6790          VBUSDATA        <^X25>,1
                  02 0165  6791          .BYTE   ^X2
                     0166  6792          VBUSDATA        <^X24>,1
                     0167  6793          VBUSDATA        <^X25>,0
                  02 0168  6794          .BYTE   ^X2
                     0169  6795          VBUSDATA        <^X24>,0
                     016A  6796          VBUSDATA        <^X25>,0
                  02 016B  6797          .BYTE   ^X2
                     016C  6798          VBUSDATA        <^X24>,0
                     016D  6799          VBUSDATA        <^X25>,0
                  02 016E  6800          .BYTE   ^X2
                     016F  6801          VBUSDATA        <^X24>,0
                     0170  6802          VBUSDATA        <^X25>,0
                  02 0171  6803          .BYTE   ^X2                             ;PROBER
                     0172  6804          VBUSDATA        <^X24>,1
                     0173  6805          VBUSDATA        <^X25>,0
                  02 0174  6806          .BYTE   ^X2
                     0175  6807          VBUSDATA        <^X24>,0
                     0176  6808          VBUSDATA        <^X25>,0
                  02 0177  6809          .BYTE   ^X2
                     0178  6810          VBUSDATA        <^X24>,0
                     0179  6811          VBUSDATA        <^X25>,0
                  02 017A  6812          .BYTE   ^X2
                     017B  6813          VBUSDATA        <^X24>,0
                     017C  6814          VBUSDATA        <^X25>,0
                  02 017D  6815          .BYTE   ^X2
                     017E  6816          VBUSDATA        <^X24>,0
                     017F  6817          VBUSDATA        <^X25>,0
                  02 0180  6818          .BYTE   ^X2                             ;PROBERW
                     0181  6819          VBUSDATA        <^X24>,1
                     0182  6820          VBUSDATA        <^X25>,0
                  02 0183  6821          .BYTE   ^X2
                     0184  6822          VBUSDATA        <^X24>,0
                     0185  6823          VBUSDATA        <^X25>,0
                  02 0186  6824          .BYTE   ^X2
                     0187  6825          VBUSDATA        <^X24>,0
                     0188  6826          VBUSDATA        <^X25>,0
                  02 0189  6827          .BYTE   ^X2
                     018A  6828          VBUSDATA        <^X24>,0
```

```
                    018B  6829          VBUSDATA        <^X25>,0
                02  018C  6830          .BYTE   ^X2
                    018D  6831          VBUSDATA        <^X24>,0
                    018E  6832          VBUSDATA        <^X25>,0
                02  018F  6833          .BYTE   ^X2                             ;INSQUE
                    0190  6834          VBUSDATA        <^X24>,0
                    0191  6835          VBUSDATA        <^X25>,0
                02  0192  6836          .BYTE   ^X2
                    0193  6837          VBUSDATA        <^X24>,0
                    0194  6838          VBUSDATA        <^X25>,0
                02  0195  6839          .BYTE   ^X2
                    0196  6840          VBUSDATA        <^X24>,0
                    0197  6841          VBUSDATA        <^X25>,0
                02  0198  6842          .BYTE   ^X2
                    0199  6843          VBUSDATA        <^X24>,0
                    019A  6844          VBUSDATA        <^X25>,0
                02  019B  6845          .BYTE   ^X2
                    019C  6846          VBUSDATA        <^X24>,0
                    019D  6847          VBUSDATA        <^X25>,0
                02  019E  6848          .BYTE   ^X2                             ;REMQUE
                    019F  6849          VBUSDATA        <^X24>,0
                    01A0  6850          VBUSDATA        <^X25>,1
                02  01A1  6851          .BYTE   ^X2
                    01A2  6852          VBUSDATA        <^X24>,0
                    01A3  6853          VBUSDATA        <^X25>,0
                02  01A4  6854          .BYTE   ^X2
                    01A5  6855          VBUSDATA        <^X24>,0
                    01A6  6856          VBUSDATA        <^X25>,0
                02  01A7  6857          .BYTE   ^X2
                    01A8  6858          VBUSDATA        <^X24>,0
                    01A9  6859          VBUSDATA        <^X25>,0
                02  01AA  6860          .BYTE   ^X2
                    01AB  6861          VBUSDATA        <^X24>,0
                    01AC  6862          VBUSDATA        <^X25>,0
                02  01AD  6863          .BYTE   ^X2                             ;BSBB
                    01AE  6864          VBUSDATA        <^X24>,0
                    01AF  6865          VBUSDATA        <^X25>,0
                02  01B0  6866          .BYTE   ^X2
                    01B1  6867          VBUSDATA        <^X24>,0
                    01B2  6868          VBUSDATA        <^X25>,0
                02  01B3  6869          .BYTE   ^X2
                    01B4  6870          VBUSDATA        <^X24>,0
                    01B5  6871          VBUSDATA        <^X25>,0
                02  01B6  6872          .BYTE   ^X2
                    01B7  6873          VBUSDATA        <^X24>,0
                    01B8  6874          VBUSDATA        <^X25>,0
                02  01B9  6875          .BYTE   ^X2
                    01BA  6876          VBUSDATA        <^X24>,0
                    01BB  6877          VBUSDATA        <^X25>,0
                02  01BC  6878          .BYTE   ^X2                             ;BRB
                    01BD  6879          VBUSDATA        <^X24>,0
                    01BE  6880          VBUSDATA        <^X25>,0
                02  01BF  6881          .BYTE   ^X2
                    01C0  6882          VBUSDATA        <^X24>,0
                    01C1  6883          VBUSDATA        <^X25>,0
```

```
                    02  01C2  6884          .BYTE   ^X2
                        01C3  6885          VBUSDATA        <^X24>,0
                        01C4  6886          VBUSDATA        <^X25>,0
                    02  01C5  6887          .BYTE   ^X2
                        01C6  6888          VBUSDATA        <^X24>,0
                        01C7  6889          VBUSDATA        <^X25>,0
                    02  01C8  6890          .BYTE   ^X2
                        01C9  6891          VBUSDATA        <^X24>,0
                        01CA  6892          VBUSDATA        <^X25>,0
                    02  01CB  6893          .BYTE   ^X2                             ;BNEQ
                        01CC  6894          VBUSDATA        <^X24>,0
                        01CD  6895          VBUSDATA        <^X25>,0
                    02  01CE  6896          .BYTE   ^X2
                        01CF  6897          VBUSDATA        <^X24>,0
                        01D0  6898          VBUSDATA        <^X25>,0
                    02  01D1  6899          .BYTE   ^X2
                        01D2  6900          VBUSDATA        <^X24>,0
                        01D3  6901          VBUSDATA        <^X25>,0
                    02  01D4  6902          .BYTE   ^X2
                        01D5  6903          VBUSDATA        <^X24>,0
                        01D6  6904          VBUSDATA        <^X25>,0
                    02  01D7  6905          .BYTE   ^X2
                        01D8  6906          VBUSDATA        <^X24>,0
                        01D9  6907          VBUSDATA        <^X25>,0
                    02  01DA  6908          .BYTE   ^X2                             ;BEQL
                        01DB  6909          VBUSDATA        <^X24>,0
                        01DC  6910          VBUSDATA        <^X25>,0
                    02  01DD  6911          .BYTE   ^X2
                        01DE  6912          VBUSDATA        <^X24>,0
                        01DF  6913          VBUSDATA        <^X25>,0
                    02  01E0  6914          .BYTE   ^X2
                        01E1  6915          VBUSDATA        <^X24>,0
                        01E2  6916          VBUSDATA        <^X25>,0
                    02  01E3  6917          .BYTE   ^X2
                        01E4  6918          VBUSDATA        <^X24>,0
                        01E5  6919          VBUSDATA        <^X25>,0
                    02  01E6  6920          .BYTE   ^X2
                        01E7  6921          VBUSDATA        <^X24>,0
                        01E8  6922          VBUSDATA        <^X25>,0
                    02  01E9  6923          .BYTE   ^X2                             ;BGTR
                        01EA  6924          VBUSDATA        <^X24>,0
                        01EB  6925          VBUSDATA        <^X25>,0
                    02  01EC  6926          .BYTE   ^X2
                        01ED  6927          VBUSDATA        <^X24>,0
                        01EE  6928          VBUSDATA        <^X25>,0
                    02  01EF  6929          .BYTE   ^X2
                        01F0  6930          VBUSDATA        <^X24>,0
                        01F1  6931          VBUSDATA        <^X25>,0
                    02  01F2  6932          .BYTE   ^X2
                        01F3  6933          VBUSDATA        <^X24>,0
                        01F4  6934          VBUSDATA        <^X25>,0
                    02  01F5  6935          .BYTE   ^X2
                        01F6  6936          VBUSDATA        <^X24>,0
                        01F7  6937          VBUSDATA        <^X25>,0
                    02  01F8  6938          .BYTE   ^X2                             ;BLEQ
```

```
                    01F9  6939          VBUSDATA        <^X24>,0
                    01FA  6940          VBUSDATA        <^X25>,0
                02  01FB  6941          .BYTE   ^X2
                    01FC  6942          VBUSDATA        <^X24>,0
                    01FD  6943          VBUSDATA        <^X25>,0
                02  01FE  6944          .BYTE   ^X2
                    01FF  6945          VBUSDATA        <^X24>,0
                    0200  6946          VBUSDATA        <^X25>,0
                02  0201  6947          .BYTE   ^X2
                    0202  6948          VBUSDATA        <^X24>,0
                    0203  6949          VBUSDATA        <^X25>,0
                02  0204  6950          .BYTE   ^X2
                    0205  6951          VBUSDATA        <^X24>,0
                    0206  6952          VBUSDATA        <^X25>,0
                02  0207  6953          .BYTE   ^X2                             ;JSB
                    0208  6954          VBUSDATA        <^X24>,0
                    0209  6955          VBUSDATA        <^X25>,0
                02  020A  6956          .BYTE   ^X2
                    020B  6957          VBUSDATA        <^X24>,0
                    020C  6958          VBUSDATA        <^X25>,0
                02  020D  6959          .BYTE   ^X2
                    020E  6960          VBUSDATA        <^X24>,0
                    020F  6961          VBUSDATA        <^X25>,0
                02  0210  6962          .BYTE   ^X2
                    0211  6963          VBUSDATA        <^X24>,0
                    0212  6964          VBUSDATA        <^X25>,0
                02  0213  6965          .BYTE   ^X2
                    0214  6966          VBUSDATA        <^X24>,0
                    0215  6967          VBUSDATA        <^X25>,0
                02  0216  6968          .BYTE   ^X2                             ;JMP
                    0217  6969          VBUSDATA        <^X24>,0
                    0218  6970          VBUSDATA        <^X25>,0
                02  0219  6971          .BYTE   ^X2
                    021A  6972          VBUSDATA        <^X24>,0
                    021B  6973          VBUSDATA        <^X25>,0
                02  021C  6974          .BYTE   ^X2
                    021D  6975          VBUSDATA        <^X24>,0
                    021E  6976          VBUSDATA        <^X25>,0
                02  021F  6977          .BYTE   ^X2
                    0220  6978          VBUSDATA        <^X24>,0
                    0221  6979          VBUSDATA        <^X25>,0
                02  0222  6980          .BYTE   ^X2
                    0223  6981          VBUSDATA        <^X24>,0
                    0224  6982          VBUSDATA        <^X25>,0
                02  0225  6983          .BYTE   ^X2                             ;BGEQ
                    0226  6984          VBUSDATA        <^X24>,0
                    0227  6985          VBUSDATA        <^X25>,0
                02  0228  6986          .BYTE   ^X2
                    0229  6987          VBUSDATA        <^X24>,0
                    022A  6988          VBUSDATA        <^X25>,0
                02  022B  6989          .BYTE   ^X2
                    022C  6990          VBUSDATA        <^X24>,0
                    022D  6991          VBUSDATA        <^X25>,0
                02  022E  6992          .BYTE   ^X2
                    022F  6993          VBUSDATA        <^X24>,0
```

```
                        0230  6994          VBUSDATA        <^X25>,0
                    02  0231  6995          .BYTE   ^X2
                        0232  6996          VBUSDATA        <^X24>,0
                        0233  6997          VBUSDATA        <^X25>,0
                    02  0234  6998          .BYTE   ^X2                             ;BLSS
                        0235  6999          VBUSDATA        <^X24>,0
                        0236  7000          VBUSDATA        <^X25>,0
                    02  0237  7001          .BYTE   ^X2
                        0238  7002          VBUSDATA        <^X24>,0
                        0239  7003          VBUSDATA        <^X25>,0
                    02  023A  7004          .BYTE   ^X2
                        023B  7005          VBUSDATA        <^X24>,0
                        023C  7006          VBUSDATA        <^X25>,0
                    02  023D  7007          .BYTE   ^X2
                        023E  7008          VBUSDATA        <^X24>,0
                        023F  7009          VBUSDATA        <^X25>,0
                    02  0240  7010          .BYTE   ^X2
                        0241  7011          VBUSDATA        <^X24>,0
                        0242  7012          VBUSDATA        <^X25>,0
                        0243  7013
                        0243  7014
                        0243  7015
                        0243  7016 BEGIN_CPU_DATA
                        0002  7017
                        0002  7018 DCS_DATA
                        0001  7019
U 00, 7700,C800,0364,0300,0470,1801     0001  7020 ;% 00   NOP
U 01, 7700,4800,5BE4,0300,2470,1802     000C  7024 ;% 01   WCTRL/LOADCRAR,WB_R[TEMP0]      ;DISABLE ^P INTERRUPT
U 02, 7700,0800,5BE4,0304,2C70,1803     0017  7028 ;% 02   WCTRL/CONWRITE,WB_R[TEMP1]      ;...
U 03, 7700,9800,D370,030F,FA70,1804     0022  7032 ;% 03   IPL_[1F]                       ;SET IPL TO 1F
U 04, 5700,4840,0364,0308,0470,1805     002D  7036 ;% 04   R[TEMP2]_RDM                   ;MOVE LOOP COUNT TO RTEMP2
U 05, 7700,D800,EF64,0301,0470,1806     0038  7040 ;% 05   PL_[2]                         ;SET PLATCH TO 2
U 06, 7700,C800,8B71,0308,0470,1807     0043  7044 ;% 06   Q_R[TEMP2].RL.P                ;ROTATE LOOP COUNT LEFT 2
U 07, 7700,4800,03A4,03D8,4870,1808     004E  7048 ;% 07   WCTRL/PC_WB,WB_Q               ;FORCE PREFETCH TO HAPPEN
U 08, 7700,4800,0364,1700,0470,1809     0059  7052 ;% 08   IRD1TEST                       ;LOAD IR REGISTER
U 09, 7600,4800,0364,0300,0470,180A     0064  7056 ;% 09   DTYPE/IDEP,STROBE.V.BUS       ;LOAD DSIZE INTO VBUS
U 0A, 7300,C800,0364,0300,0470,180B     006F  7060 ;% 0A   NOP,STOP.CLOCK                ;STOP THE CPU CLOCK
U 0B, 7700,4800,0364,0300,0470,180C     007A  7064 ;% 0B   NOP
U 0C, 7700,C800,0364,1B00,0470,180D     0085  7068 ;% 0C   LOD INC BRA?                  ;INCREMENT THE IRD COUNTER
U 0D, 7600,C800,0364,0300,0470,180E     0090  7072 ;% 0D   DTYPE/IDEP,STROBE.V.BUS       ;LOAD DSIZE INTO VBUS
U 0E, 7300,4800,0364,0300,0470,180F     009B  7076 ;% 0E   NOP,STOP.CLOCK                ;STOP THE CPU CLOCK
                        00A6  7080
                        00A6  7081 RTEMP_DATA
                        0001  7082
                    00C00000  0001  7083          .LONG   ^X00C00000                  ;DISABLE ^P INTERRUPT
                    00400000  0005  7084          .LONG   ^X00400000                  ;...
                        0009  7085
                        0009  7086 CACHE_DATA
                        0001  7087
                    00000001  0001  7088          .LONG   ^X00000001                  ;NOP
                    00000002  0005  7089          .LONG   ^X00000002                  ;REI
                    00000003  0009  7090          .LONG   ^X00000003                  ;BPT
                    00000004  000D  7091          .LONG   ^X00000004                  ;RET
                    00000005  0011  7092          .LONG   ^X00000005                  ;RSB
                    00000006  0015  7093          .LONG   ^X00000006                  ;LDPCTX
```

```
00000007  0019  7094          .LONG   ^X00000007                      ;SVPCTX
00000008  001D  7095          .LONG   ^X00000008                      ;CVTPS
00000009  0021  7096          .LONG   ^X00000009                      ;CVTSP
0000000A  0025  7097          .LONG   ^X0000000A                      ;INDEX
0000000B  0029  7098          .LONG   ^X0000000B                      ;CRC
0000000C  002D  7099          .LONG   ^X0000000C                      ;PROBER
0000000D  0031  7100          .LONG   ^X0000000D                      ;PROBEW
0000000E  0035  7101          .LONG   ^X0000000E                      ;INSQUE
0000000F  0039  7102          .LONG   ^X0000000F                      ;REMQUE
00000010  003D  7103          .LONG   ^X00000010                      ;BSBB
00000011  0041  7104          .LONG   ^X00000011                      ;BRB
00000012  0045  7105          .LONG   ^X00000012                      ;BNEQ
00000013  0049  7106          .LONG   ^X00000013                      ;BEQL
00000014  004D  7107          .LONG   ^X00000014                      ;BGTR
00000015  0051  7108          .LONG   ^X00000015                      ;BLEQ
00000016  0055  7109          .LONG   ^X00000016                      ;JSB
00000017  0059  7110          .LONG   ^X00000017                      ;JMP
00000018  005D  7111          .LONG   ^X00000018                      ;BGEQ
00000019  0061  7112          .LONG   ^X00000019                      ;BLSS
00000000  0065  7113          .LONG   ^X00000000
          0069  7114
          0069  7115 END_CPU_DATA
          0243  7116
          0243  7117
          0243  7118 END_TEST_DATA
          0243
          0243      ;-----------------------------------------------------------------------------
          0243  7119 ENDTEST
```

```
                0000  7120
                0021        .SBTTL  "TEST   085     NATIVE MODE DSIZE ROM TEST 3
                0021  7122 ;*******************************************************************************
                0021  7123 ;++
                0021  7124 ;
                0021  7125 ; FUNCTIONAL DESCRIPTION:
                0021  7126 ;
                0021  7127 ;       THIS IS ONE IN A SERIES OF 14 TESTS THAT CHECK THE CONTENTS OF THE
                0021  7128 ;       DSIZE ROM.
                0021  7129 ;
                0021  7130 ;
                0021  7131 ; TEST ALGORITHM:
                0021  7132 ;
                0021  7133 ;       1.  CREATE AN INDEX (K) TO INDEX RESULTS TABLE
                0021  7134 ;       2.  CREATE LOOP (I) OF 25
                0021  7135 ;       3.  SAVE LOOP COUNT IN 1$
                0021  7136 ;       4.  MOVE LOOP COUNT TO WDREG
                0021  7137 ;       5.  EXECUTE DCS PROGRAM
                0021  7138 ;               a.  DISABLE ^P INTERRUPTS
                0021  7139 ;               b.  SET IPL TO 1F
                0021  7140 ;               c.  MOVE LOOP COUNT TO RTEMP2
                0021  7141 ;               d.  ROTATE LOOP COUNT LEFT 2 TO FORM CACHE ADDRESS
                0021  7142 ;               e.  LOAD THIS ADDRESS INTO PC TO CAUSE PREFETCH
                0021  7143 ;               f.  PERFORM IRD1TEST TO CAUSE IR TO LOAD INSTRUCTION
                0021  7144 ;               g.  SET DTYPE TO INSTURCTION DEPENDENT AND STROBE THE VBUS
                0021  7145 ;       6.  TEST VBUS FOR CORRECT DSIZE
                0021  7146 ;       7.  IF NO ERROR CREATE LOOP (J) OF 5
                0021  7147 ;       8.  EXECUTE DCS PROGRAM
                0021  7148 ;               a.  INCREMENT THE IRD COUNTER
                0021  7149 ;               b.  SET DTYPE TO INSTRUCTION DEPENDENT AND STROBE THE VBUS
                0021  7150 ;       9.  TEST VBUS FOR CORRECT DSIZE
                0021  7151 ;      10.  INCREMENT INDEX K BY ONE
                0021  7152 ;      11.  IF NO ERROR GO TO STEP 8 UNTIL LOOP J IS EXHAUSTED
                0021  7153 ;      12.  GO TO STEP 3 UNTIL LOOP I IS EXHAUSTED
                0021  7154 ;
                0021  7155 ;
                0021  7156 ; TEST PATTERNS:
                0021  7157 ;
                0021  7158 ;       LOOP COUNT I            INSTRUCTION
                0021  7159 ;            1                     BGTRU
                0021  7160 ;            2                     BLEQU
                0021  7161 ;            3                     BVC
                0021  7162 ;            4                     BVS
                0021  7163 ;            5                     BGEQU
                0021  7164 ;            6                     BLSSU
                0021  7165 ;            7                     ADDP4
                0021  7166 ;            8                     ADDP6
                0021  7167 ;            9                     SUBP4
                0021  7168 ;            A                     SUBP6
                0021  7169 ;            B                     CVTPT
                0021  7170 ;            C                     MULP
                0021  7171 ;            D                     CVTTP
                0021  7172 ;            E                     DIVP
                0021  7173 ;            F                     MOVC3
                0021  7174 ;           10                     CMPC3
```

```
0021  7175 ;           11                      SCANC
0021  7176 ;           12                      SPANC
0021  7177 ;           13                      MOVC5
0021  7178 ;           14                      CMPC5
0021  7179 ;           15                      MOVTC
0021  7180 ;           16                      MOVTUC
0021  7181 ;           17                      BSBW
0021  7182 ;           18                      BRW
0021  7183 ;           19                      CVTWL
0021  7184 ;
0021  7185 ;
0021  7186 ; LOGIC DESCRIPTION:
0021  7187 ;
0021  7188 ;       THIS TEST ALONG WITH THE OTHER 13 TESTS IN THIS SERIES CHECK ALL THE
0021  7189 ;       LOCATIONS BLASTED INTO THE DSIZE ROM.  ANY FALIURES SHOULD BE CONFINED
0021  7190 ;       TO THE ROM.
0021  7191 ;
0021  7192 ;
0021  7193 ; ASSUMPTIONS:
0021  7194 ;
0021  7195 ;       THIS TEST ASSUMES THE PREFETCH LOGIC AND INSTRUCTION REGISTER ARE
0021  7196 ;       OPERATIONAL.
0021  7197 ;
0021  7198 ;
0021  7199 ; ERROR DESCRIPTION:
0021  7200 ;
0021  7201 ;       EXPECTED VBUS DATA (DSIZE)
0021  7202 ;       RECEIVED VBUS DATA
0021  7203 ;       LOOP COUNT I
0021  7204 ;       LOOP COUNT J
0021  7205 ;       LOOP COUNT K
0021  7206 ;               (BITS <7:0> = VBUS BIT POSITION)
0021  7207 ;               (BITS <8> = BIT STATE)
0021  7208 ;
0021  7209 ;--
0021  7210 ;*******************************************************************************
0021  7211
0021  7212
0021  7213         INITIALIZE                              ;INIT THE CPU
0023  7214         LDINDEX         K,1                     ;INIT INDEX K TO 1
002A  7215         LOOP            I,1,25                  ;CREATE A TEST LOOP OF 50
0031  7216         SAVEINDEX       I,1$                    ;SAVE LOOP COUNT IN 1$
0036  7217         LOADREG         WDREG,1$,,L             ;LOAD LOOP COUNT INTO WDREG
003E  7218         ERRLOOP                                 ;ERROR LOOP POINT
0040  7219         FETCH           0                       ;START DCS PROGRAM
0045  7220         BRSTCLK         <^X0A>,INH              ;END DCS PROGRAM
0049  7221         CMPVBUS         2$,I                    ;TEST DSIZE
004E  7222         IFERROR                                 ;ROM FAILURE
0053  7223         LOOP            J,1,5                   ;CREATE TEST LOOP OF 5
005A  7224         FETCH           <^X0B>                  ;START DCS PROGRAM
005F  7225         BRSTCLK         <^X0E>,INH              ;END DCS PROGRAM
0063  7226         CMPVBUS         3$,K                    ;TEST DSIZE
0068  7227         IFERROR                                 ;ROM FAILURE
006D  7228         INCINDEX        K                       ;INCREMENT INDEX K BY ONE
0070  7229         ENDLOOP         J                       ;END TEST LOOP J
```

```
                     0073  7230          ENDLOOP         I                       ;END TEST LOOP I
                     0076  7231          SKIP                                    ;GO TO NEXT TEST
                     007D  7232
                     007D  7233
                     007D  7234 BEGIN_TEST_DATA
                     007D
                     007D          ;-------------------------------------------------------------------------
                     007D  7235
                00000000  007D  7236 1$:      .LONG    ^X00000000                ;LOOP COUNT STORAGE
                       02  0081  7237 2$:      .BYTE    ^X2                       ;RESULTS TABLE (IRD 0)
                           0082  7238          VBUSDATA         <^X24>,0          ;BGTRU
                           0083  7239          VBUSDATA         <^X25>,0
                       02  0084  7240          .BYTE    ^X2                       ;BLEQU
                           0085  7241          VBUSDATA         <^X24>,0
                           0086  7242          VBUSDATA         <^X25>,0
                       02  0087  7243          .BYTE    ^X2                       ;BVC
                           0088  7244          VBUSDATA         <^X24>,0
                           0089  7245          VBUSDATA         <^X25>,0
                       02  008A  7246          .BYTE    ^X2                       ;BVS
                           008B  7247          VBUSDATA         <^X24>,0
                           008C  7248          VBUSDATA         <^X25>,0
                       02  008D  7249          .BYTE    ^X2                       ;BGEQU
                           008E  7250          VBUSDATA         <^X24>,0
                           008F  7251          VBUSDATA         <^X25>,0
                       02  0090  7252          .BYTE    ^X2                       ;BLSSU
                           0091  7253          VBUSDATA         <^X24>,0
                           0092  7254          VBUSDATA         <^X25>,0
                       02  0093  7255          .BYTE    ^X2                       ;ADDP4
                           0094  7256          VBUSDATA         <^X24>,1
                           0095  7257          VBUSDATA         <^X25>,0
                       02  0096  7258          .BYTE    ^X2                       ;ADDP6
                           0097  7259          VBUSDATA         <^X24>,1
                           0098  7260          VBUSDATA         <^X25>,0
                       02  0099  7261          .BYTE    ^X2                       ;SUBP4
                           009A  7262          VBUSDATA         <^X24>,1
                           009B  7263          VBUSDATA         <^X25>,0
                       02  009C  7264          .BYTE    ^X2                       ;SUBP6
                           009D  7265          VBUSDATA         <^X24>,1
                           009E  7266          VBUSDATA         <^X25>,0
                       02  009F  7267          .BYTE    ^X2                       ;CVTPT
                           00A0  7268          VBUSDATA         <^X24>,1
                           00A1  7269          VBUSDATA         <^X25>,0
                       02  00A2  7270          .BYTE    ^X2                       ;MULP
                           00A3  7271          VBUSDATA         <^X24>,1
                           00A4  7272          VBUSDATA         <^X25>,0
                       02  00A5  7273          .BYTE    ^X2                       ;CVTTP
                           00A6  7274          VBUSDATA         <^X24>,1
                           00A7  7275          VBUSDATA         <^X25>,0
                       02  00A8  7276          .BYTE    ^X2                       ;DIVP
                           00A9  7277          VBUSDATA         <^X24>,1
                           00AA  7278          VBUSDATA         <^X25>,0
                       02  00AB  7279          .BYTE    ^X2                       ;MOVC3
                           00AC  7280          VBUSDATA         <^X24>,1
                           00AD  7281          VBUSDATA         <^X25>,0
                       02  00AE  7282          .BYTE    ^X2                       ;CMPC3
```

```
                    00AF  7283          VBUSDATA        <^X24>,1
                    00B0  7284          VBUSDATA        <^X25>,0
                 02 00B1  7285          .BYTE   ^X2                             ;SCANC
                    00B2  7286          VBUSDATA        <^X24>,1
                    00B3  7287          VBUSDATA        <^X25>,0
                 02 00B4  7288          .BYTE   ^X2                             ;SPANC
                    00B5  7289          VBUSDATA        <^X24>,1
                    00B6  7290          VBUSDATA        <^X25>,0
                 02 00B7  7291          .BYTE   ^X2                             ;MOVC5
                    00B8  7292          VBUSDATA        <^X24>,1
                    00B9  7293          VBUSDATA        <^X25>,0
                 02 00BA  7294          .BYTE   ^X2                             ;CMPC5
                    00BB  7295          VBUSDATA        <^X24>,1
                    00BC  7296          VBUSDATA        <^X25>,0
                 02 00BD  7297          .BYTE   ^X2                             ;MOVTC
                    00BE  7298          VBUSDATA        <^X24>,1
                    00BF  7299          VBUSDATA        <^X25>,0
                 02 00C0  7300          .BYTE   ^X2                             ;MOVTUC
                    00C1  7301          VBUSDATA        <^X24>,1
                    00C2  7302          VBUSDATA        <^X25>,0
                 02 00C3  7303          .BYTE   ^X2                             ;BSBW
                    00C4  7304          VBUSDATA        <^X24>,1
                    00C5  7305          VBUSDATA        <^X25>,0
                 02 00C6  7306          .BYTE   ^X2                             ;BRW
                    00C7  7307          VBUSDATA        <^X24>,1
                    00C8  7308          VBUSDATA        <^X25>,0
                 02 00C9  7309          .BYTE   ^X2                             ;CVTWL
                    00CA  7310          VBUSDATA        <^X24>,1
                    00CB  7311          VBUSDATA        <^X25>,0
                    00CC  7312 ...
                 02 00CC  7313 3$:      .BYTE   ^X2                             ;RESULTS TABLE (IRD 1 TO 5)
                    00CD  7314          VBUSDATA        <^X24>,0                ;BGTRU
                    00CE  7315          VBUSDATA        <^X25>,0
                 02 00CF  7316          .BYTE   ^X2
                    00D0  7317          VBUSDATA        <^X24>,0
                    00D1  7318          VBUSDATA        <^X25>,0
                 02 00D2  7319          .BYTE   ^X2
                    00D3  7320          VBUSDATA        <^X24>,0
                    00D4  7321          VBUSDATA        <^X25>,0
                 02 00D5  7322          .BYTE   ^X2
                    00D6  7323          VBUSDATA        <^X24>,0
                    00D7  7324          VBUSDATA        <^X25>,0
                 02 00D8  7325          .BYTE   ^X2
                    00D9  7326          VBUSDATA        <^X24>,0
                    00DA  7327          VBUSDATA        <^X25>,0
                 02 00DB  7328          .BYTE   ^X2                             ;BLEQU
                    00DC  7329          VBUSDATA        <^X24>,0
                    00DD  7330          VBUSDATA        <^X25>,0
                 02 00DE  7331          .BYTE   ^X2
                    00DF  7332          VBUSDATA        <^X24>,0
                    00E0  7333          VBUSDATA        <^X25>,0
                 02 00E1  7334          .BYTE   ^X2
                    00E2  7335          VBUSDATA        <^X24>,0
                    00E3  7336          VBUSDATA        <^X25>,0
                 02 00E4  7337          .BYTE   ^X2
```

```
                    00E5  7338          VBUSDATA        <^X24>,0
                    00E6  7339          VBUSDATA        <^X25>,0
                02  00E7  7340          .BYTE   ^X2
                    00E8  7341          VBUSDATA        <^X24>,0
                    00E9  7342          VBUSDATA        <^X25>,0
                02  00EA  7343          .BYTE   ^X2                             ;BVC
                    00EB  7344          VBUSDATA        <^X24>,0
                    00EC  7345          VBUSDATA        <^X25>,0
                02  00ED  7346          .BYTE   ^X2
                    00EE  7347          VBUSDATA        <^X24>,0
                    00EF  7348          VBUSDATA        <^X25>,0
                02  00F0  7349          .BYTE   ^X2
                    00F1  7350          VBUSDATA        <^X24>,0
                    00F2  7351          VBUSDATA        <^X25>,0
                02  00F3  7352          .BYTE   ^X2
                    00F4  7353          VBUSDATA        <^X24>,0
                    00F5  7354          VBUSDATA        <^X25>,0
                02  00F6  7355          .BYTE   ^X2
                    00F7  7356          VBUSDATA        <^X24>,0
                    00F8  7357          VBUSDATA        <^X25>,0
                02  00F9  7358          .BYTE   ^X2                             ;BVS
                    00FA  7359          VBUSDATA        <^X24>,0
                    00FB  7360          VBUSDATA        <^X25>,0
                02  00FC  7361          .BYTE   ^X2
                    00FD  7362          VBUSDATA        <^X24>,0
                    00FE  7363          VBUSDATA        <^X25>,0
                02  00FF  7364          .BYTE   ^X2
                    0100  7365          VBUSDATA        <^X24>,0
                    0101  7366          VBUSDATA        <^X25>,0
                02  0102  7367          .BYTE   ^X2
                    0103  7368          VBUSDATA        <^X24>,0
                    0104  7369          VBUSDATA        <^X25>,0
                02  0105  7370          .BYTE   ^X2
                    0106  7371          VBUSDATA        <^X24>,0
                    0107  7372          VBUSDATA        <^X25>,0
                02  0108  7373          .BYTE   ^X2                             ;BGEQU
                    0109  7374          VBUSDATA        <^X24>,0
                    010A  7375          VBUSDATA        <^X25>,0
                02  010B  7376          .BYTE   ^X2
                    010C  7377          VBUSDATA        <^X24>,0
                    010D  7378          VBUSDATA        <^X25>,0
                02  010E  7379          .BYTE   ^X2
                    010F  7380          VBUSDATA        <^X24>,0
                    0110  7381          VBUSDATA        <^X25>,0
                02  0111  7382          .BYTE   ^X2
                    0112  7383          VBUSDATA        <^X24>,0
                    0113  7384          VBUSDATA        <^X25>,0
                02  0114  7385          .BYTE   ^X2
                    0115  7386          VBUSDATA        <^X24>,0
                    0116  7387          VBUSDATA        <^X25>,0
                02  0117  7388          .BYTE   ^X2                             ;BLSSU
                    0118  7389          VBUSDATA        <^X24>,0
                    0119  7390          VBUSDATA        <^X25>,0
                02  011A  7391          .BYTE   ^X2
                    011B  7392          VBUSDATA        <^X24>,0
```

```
                    011C  7393          VBUSDATA        <^X25>,0
                02  011D  7394          .BYTE   ^X2
                    011E  7395          VBUSDATA        <^X24>,0
                    011F  7396          VBUSDATA        <^X25>,0
                02  0120  7397          .BYTE   ^X2
                    0121  7398          VBUSDATA        <^X24>,0
                    0122  7399          VBUSDATA        <^X25>,0
                02  0123  7400          .BYTE   ^X2
                    0124  7401          VBUSDATA        <^X24>,0
                    0125  7402          VBUSDATA        <^X25>,0
                02  0126  7403          .BYTE   ^X2                             ;ADDP4
                    0127  7404          VBUSDATA        <^X24>,0
                    0128  7405          VBUSDATA        <^X25>,0
                02  0129  7406          .BYTE   ^X2
                    012A  7407          VBUSDATA        <^X24>,1
                    012B  7408          VBUSDATA        <^X25>,0
                02  012C  7409          .BYTE   ^X2
                    012D  7410          VBUSDATA        <^X24>,0
                    012E  7411          VBUSDATA        <^X25>,0
                02  012F  7412          .BYTE   ^X2
                    0130  7413          VBUSDATA        <^X24>,0
                    0131  7414          VBUSDATA        <^X25>,0
                02  0132  7415          .BYTE   ^X2
                    0133  7416          VBUSDATA        <^X24>,0
                    0134  7417          VBUSDATA        <^X25>,0
                02  0135  7418          .BYTE   ^X2                             ;ADDP6
                    0136  7419          VBUSDATA        <^X24>,0
                    0137  7420          VBUSDATA        <^X25>,0
                02  0138  7421          .BYTE   ^X2
                    0139  7422          VBUSDATA        <^X24>,1
                    013A  7423          VBUSDATA        <^X25>,0
                02  013B  7424          .BYTE   ^X2
                    013C  7425          VBUSDATA        <^X24>,0
                    013D  7426          VBUSDATA        <^X25>,0
                02  013E  7427          .BYTE   ^X2
                    013F  7428          VBUSDATA        <^X24>,1
                    0140  7429          VBUSDATA        <^X25>,0
                02  0141  7430          .BYTE   ^X2
                    0142  7431          VBUSDATA        <^X24>,0
                    0143  7432          VBUSDATA        <^X25>,0
                02  0144  7433          .BYTE   ^X2                             ;SUBP4
                    0145  7434          VBUSDATA        <^X24>,0
                    0146  7435          VBUSDATA        <^X25>,0
                02  0147  7436          .BYTE   ^X2
                    0148  7437          VBUSDATA        <^X24>,1
                    0149  7438          VBUSDATA        <^X25>,0
                02  014A  7439          .BYTE   ^X2
                    014B  7440          VBUSDATA        <^X24>,0
                    014C  7441          VBUSDATA        <^X25>,0
                02  014D  7442          .BYTE   ^X2
                    014E  7443          VBUSDATA        <^X24>,0
                    014F  7444          VBUSDATA        <^X25>,0
                02  0150  7445          .BYTE   ^X2
                    0151  7446          VBUSDATA        <^X24>,0
                    0152  7447          VBUSDATA        <^X25>,0
```

```
02  0153  7448          .BYTE   ^X2                                  ;SUBP6
    0154  7449          VBUSDATA        <^X24>,0
    0155  7450          VBUSDATA        <^X25>,0
02  0156  7451          .BYTE   ^X2
    0157  7452          VBUSDATA        <^X24>,1
    0158  7453          VBUSDATA        <^X25>,0
02  0159  7454          .BYTE   ^X2
    015A  7455          VBUSDATA        <^X24>,0
    015B  7456          VBUSDATA        <^X25>,0
02  015C  7457          .BYTE   ^X2
    015D  7458          VBUSDATA        <^X24>,1
    015E  7459          VBUSDATA        <^X25>,0
02  015F  7460          .BYTE   ^X2
    0160  7461          VBUSDATA        <^X24>,0
    0161  7462          VBUSDATA        <^X25>,0
02  0162  7463          .BYTE   ^X2                                  ;CVTPT
    0163  7464          VBUSDATA        <^X24>,0
    0164  7465          VBUSDATA        <^X25>,0
02  0165  7466          .BYTE   ^X2
    0166  7467          VBUSDATA        <^X24>,0
    0167  7468          VBUSDATA        <^X25>,0
02  0168  7469          .BYTE   ^X2
    0169  7470          VBUSDATA        <^X24>,1
    016A  7471          VBUSDATA        <^X25>,0
02  016B  7472          .BYTE   ^X2
    016C  7473          VBUSDATA        <^X24>,0
    016D  7474          VBUSDATA        <^X25>,0
02  016E  7475          .BYTE   ^X2
    016F  7476          VBUSDATA        <^X24>,0
    0170  7477          VBUSDATA        <^X25>,0
02  0171  7478          .BYTE   ^X2                                  ;MULP
    0172  7479          VBUSDATA        <^X24>,0
    0173  7480          VBUSDATA        <^X25>,0
02  0174  7481          .BYTE   ^X2
    0175  7482          VBUSDATA        <^X24>,1
    0176  7483          VBUSDATA        <^X25>,0
02  0177  7484          .BYTE   ^X2
    0178  7485          VBUSDATA        <^X24>,0
    0179  7486          VBUSDATA        <^X25>,0
02  017A  7487          .BYTE   ^X2
    017B  7488          VBUSDATA        <^X24>,1
    017C  7489          VBUSDATA        <^X25>,0
02  017D  7490          .BYTE   ^X2
    017E  7491          VBUSDATA        <^X24>,0
    017F  7492          VBUSDATA        <^X25>,0
02  0180  7493          .BYTE   ^X2                                  ;CVTTP
    0181  7494          VBUSDATA        <^X24>,0
    0182  7495          VBUSDATA        <^X25>,0
02  0183  7496          .BYTE   ^X2
    0184  7497          VBUSDATA        <^X24>,0
    0185  7498          VBUSDATA        <^X25>,0
02  0186  7499          .BYTE   ^X2
    0187  7500          VBUSDATA        <^X24>,1
    0188  7501          VBUSDATA        <^X25>,0
02  0189  7502          .BYTE   ^X2
```

```
                    018A  7503          VBUSDATA        <^X24>,0
                    018B  7504          VBUSDATA        <^X25>,0
                02  018C  7505          .BYTE   ^X2
                    018D  7506          VBUSDATA        <^X24>,0
                    018E  7507          VBUSDATA        <^X25>,0
                02  018F  7508          .BYTE   ^X2                             ;DIVP
                    0190  7509          VBUSDATA        <^X24>,0
                    0191  7510          VBUSDATA        <^X25>,0
                02  0192  7511          .BYTE   ^X2
                    0193  7512          VBUSDATA        <^X24>,1
                    0194  7513          VBUSDATA        <^X25>,0
                02  0195  7514          .BYTE   ^X2
                    0196  7515          VBUSDATA        <^X24>,0
                    0197  7516          VBUSDATA        <^X25>,0
                02  0198  7517          .BYTE   ^X2
                    0199  7518          VBUSDATA        <^X24>,1
                    019A  7519          VBUSDATA        <^X25>,0
                02  019B  7520          .BYTE   ^X2
                    019C  7521          VBUSDATA        <^X24>,0
                    019D  7522          VBUSDATA        <^X25>,0
                02  019E  7523          .BYTE   ^X2                             ;MOVC3
                    019F  7524          VBUSDATA        <^X24>,0
                    01A0  7525          VBUSDATA        <^X25>,0
                02  01A1  7526          .BYTE   ^X2
                    01A2  7527          VBUSDATA        <^X24>,0
                    01A3  7528          VBUSDATA        <^X25>,0
                02  01A4  7529          .BYTE   ^X2
                    01A5  7530          VBUSDATA        <^X24>,0
                    01A6  7531          VBUSDATA        <^X25>,0
                02  01A7  7532          .BYTE   ^X2
                    01A8  7533          VBUSDATA        <^X24>,0
                    01A9  7534          VBUSDATA        <^X25>,0
                02  01AA  7535          .BYTE   ^X2
                    01AB  7536          VBUSDATA        <^X24>,0
                    01AC  7537          VBUSDATA        <^X25>,0
                02  01AD  7538          .BYTE   ^X2                             ;CMPC3
                    01AE  7539          VBUSDATA        <^X24>,0
                    01AF  7540          VBUSDATA        <^X25>,0
                02  01B0  7541          .BYTE   ^X2
                    01B1  7542          VBUSDATA        <^X24>,0
                    01B2  7543          VBUSDATA        <^X25>,0
                02  01B3  7544          .BYTE   ^X2
                    01B4  7545          VBUSDATA        <^X24>,0
                    01B5  7546          VBUSDATA        <^X25>,0
                02  01B6  7547          .BYTE   ^X2
                    01B7  7548          VBUSDATA        <^X24>,0
                    01B8  7549          VBUSDATA        <^X25>,0
                02  01B9  7550          .BYTE   ^X2
                    01BA  7551          VBUSDATA        <^X24>,0
                    01BB  7552          VBUSDATA        <^X25>,0
                02  01BC  7553          .BYTE   ^X2                             ;SCANC
                    01BD  7554          VBUSDATA        <^X24>,0
                    01BE  7555          VBUSDATA        <^X25>,0
                02  01BF  7556          .BYTE   ^X2
                    01C0  7557          VBUSDATA        <^X24>,0
```

```
                    01C1  7558          VBUSDATA        <^X25>,0
                02  01C2  7559          .BYTE   ^X2
                    01C3  7560          VBUSDATA        <^X24>,0
                    01C4  7561          VBUSDATA        <^X25>,0
                02  01C5  7562          .BYTE   ^X2
                    01C6  7563          VBUSDATA        <^X24>,0
                    01C7  7564          VBUSDATA        <^X25>,0
                02  01C8  7565          .BYTE   ^X2
                    01C9  7566          VBUSDATA        <^X24>,0
                    01CA  7567          VBUSDATA        <^X25>,0
                02  01CB  7568          .BYTE   ^X2                             ;SPANC
                    01CC  7569          VBUSDATA        <^X24>,0
                    01CD  7570          VBUSDATA        <^X25>,0
                02  01CE  7571          .BYTE   ^X2
                    01CF  7572          VBUSDATA        <^X24>,0
                    01D0  7573          VBUSDATA        <^X25>,0
                02  01D1  7574          .BYTE   ^X2
                    01D2  7575          VBUSDATA        <^X24>,0
                    01D3  7576          VBUSDATA        <^X25>,0
                02  01D4  7577          .BYTE   ^X2
                    01D5  7578          VBUSDATA        <^X24>,0
                    01D6  7579          VBUSDATA        <^X25>,0
                02  01D7  7580          .BYTE   ^X2
                    01D8  7581          VBUSDATA        <^X24>,0
                    01D9  7582          VBUSDATA        <^X25>,0
                02  01DA  7583          .BYTE   ^X2                             ;MOVC5
                    01DB  7584          VBUSDATA        <^X24>,0
                    01DC  7585          VBUSDATA        <^X25>,0
                02  01DD  7586          .BYTE   ^X2
                    01DE  7587          VBUSDATA        <^X24>,0
                    01DF  7588          VBUSDATA        <^X25>,0
                02  01E0  7589          .BYTE   ^X2
                    01E1  7590          VBUSDATA        <^X24>,1
                    01E2  7591          VBUSDATA        <^X25>,0
                02  01E3  7592          .BYTE   ^X2
                    01E4  7593          VBUSDATA        <^X24>,0
                    01E5  7594          VBUSDATA        <^X25>,0
                02  01E6  7595          .BYTE   ^X2
                    01E7  7596          VBUSDATA        <^X24>,0
                    01E8  7597          VBUSDATA        <^X25>,0
                02  01E9  7598          .BYTE   ^X2                             ;CMPC5
                    01EA  7599          VBUSDATA        <^X24>,0
                    01EB  7600          VBUSDATA        <^X25>,0
                02  01EC  7601          .BYTE   ^X2
                    01ED  7602          VBUSDATA        <^X24>,0
                    01EE  7603          VBUSDATA        <^X25>,0
                02  01EF  7604          .BYTE   ^X2
                    01F0  7605          VBUSDATA        <^X24>,1
                    01F1  7606          VBUSDATA        <^X25>,0
                02  01F2  7607          .BYTE   ^X2
                    01F3  7608          VBUSDATA        <^X24>,0
                    01F4  7609          VBUSDATA        <^X25>,0
                02  01F5  7610          .BYTE   ^X2
                    01F6  7611          VBUSDATA        <^X24>,0
                    01F7  7612          VBUSDATA        <^X25>,0
```

```
                    02  01F8  7613          .BYTE   ^X2                                 ;MOVTC
                        01F9  7614          VBUSDATA        <^X24>,0
                        01FA  7615          VBUSDATA        <^X25>,0
                    02  01FB  7616          .BYTE   ^X2
                        01FC  7617          VBUSDATA        <^X24>,0
                        01FD  7618          VBUSDATA        <^X25>,0
                    02  01FE  7619          .BYTE   ^X2
                        01FF  7620          VBUSDATA        <^X24>,0
                        0200  7621          VBUSDATA        <^X25>,0
                    02  0201  7622          .BYTE   ^X2
                        0202  7623          VBUSDATA        <^X24>,1
                        0203  7624          VBUSDATA        <^X25>,0
                    02  0204  7625          .BYTE   ^X2
                        0205  7626          VBUSDATA        <^X24>,0
                        0206  7627          VBUSDATA        <^X25>,0
                    02  0207  7628          .BYTE   ^X2                                 ;MOVTUC
                        0208  7629          VBUSDATA        <^X24>,0
                        0209  7630          VBUSDATA        <^X25>,0
                    02  020A  7631          .BYTE   ^X2
                        020B  7632          VBUSDATA        <^X24>,0
                        020C  7633          VBUSDATA        <^X25>,0
                    02  020D  7634          .BYTE   ^X2
                        020E  7635          VBUSDATA        <^X24>,0
                        020F  7636          VBUSDATA        <^X25>,0
                    02  0210  7637          .BYTE   ^X2
                        0211  7638          VBUSDATA        <^X24>,1
                        0212  7639          VBUSDATA        <^X25>,0
                    02  0213  7640          .BYTE   ^X2
                        0214  7641          VBUSDATA        <^X24>,0
                        0215  7642          VBUSDATA        <^X25>,0
                    02  0216  7643          .BYTE   ^X2                                 ;BSBW
                        0217  7644          VBUSDATA        <^X24>,0
                        0218  7645          VBUSDATA        <^X25>,0
                    02  0219  7646          .BYTE   ^X2
                        021A  7647          VBUSDATA        <^X24>,0
                        021B  7648          VBUSDATA        <^X25>,0
                    02  021C  7649          .BYTE   ^X2
                        021D  7650          VBUSDATA        <^X24>,0
                        021E  7651          VBUSDATA        <^X25>,0
                    02  021F  7652          .BYTE   ^X2
                        0220  7653          VBUSDATA        <^X24>,0
                        0221  7654          VBUSDATA        <^X25>,0
                    02  0222  7655          .BYTE   ^X2
                        0223  7656          VBUSDATA        <^X24>,0
                        0224  7657          VBUSDATA        <^X25>,0
                    02  0225  7658          .BYTE   ^X2                                 ;BRW
                        0226  7659          VBUSDATA        <^X24>,0
                        0227  7660          VBUSDATA        <^X25>,0
                    02  0228  7661          .BYTE   ^X2
                        0229  7662          VBUSDATA        <^X24>,0
                        022A  7663          VBUSDATA        <^X25>,0
                    02  022B  7664          .BYTE   ^X2
                        022C  7665          VBUSDATA        <^X24>,0
                        022D  7666          VBUSDATA        <^X25>,0
                    02  022E  7667          .BYTE   ^X2
```

```
                                 022F  7668          VBUSDATA        <^X24>,0
                                 0230  7669          VBUSDATA        <^X25>,0
                              02 0231  7670          .BYTE   ^X2
                                 0232  7671          VBUSDATA        <^X24>,0
                                 0233  7672          VBUSDATA        <^X25>,0
                              02 0234  7673          .BYTE   ^X2                             ;CVTWL
                                 0235  7674          VBUSDATA        <^X24>,0
                                 0236  7675          VBUSDATA        <^X25>,1
                              02 0237  7676          .BYTE   ^X2
                                 0238  7677          VBUSDATA        <^X24>,0
                                 0239  7678          VBUSDATA        <^X25>,0
                              02 023A  7679          .BYTE   ^X2
                                 023B  7680          VBUSDATA        <^X24>,0
                                 023C  7681          VBUSDATA        <^X25>,0
                              02 023D  7682          .BYTE   ^X2
                                 023E  7683          VBUSDATA        <^X24>,0
                                 023F  7684          VBUSDATA        <^X25>,0
                              02 0240  7685          .BYTE   ^X2
                                 0241  7686          VBUSDATA        <^X24>,0
                                 0242  7687          VBUSDATA        <^X25>,0
                                 0243  7688
                                 0243  7689
                                 0243  7690
                                 0243  7691 BEGIN_CPU_DATA
                                 0002  7692
                                 0002  7693 DCS_DATA
                                 0001  7694
U 00, 7700,C800,0364,0300,0470,1801 0001  7695 ;% 00    NOP
U 01, 7700,4800,5BE4,0300,2470,1802 000C  7699 ;% 01    WCTRL/LOADCRAR,WB_R[TEMP0]              ;DISABLE ^P INTERRUPT
U 02, 7700,0800,5BE4,0304,2C70,1803 0017  7703 ;% 02    WCTRL/CONWRITE,WB_R[TEMP1]              ;...
U 03, 7700,9800,D370,030F,FA70,1804 0022  7707 ;% 03    IPL_[1F]                                ;SET IPL TO 1F
U 04, 5700,4840,0364,0308,0470,1805 002D  7711 ;% 04    R[TEMP2]_RDM                            ;MOVE LOOP COUNT TO RTEMP2
U 05, 7700,D800,EF64,0301,0470,1806 0038  7715 ;% 05    PL_[2]                                  ;SET PLATCH TO 2
U 06, 7700,C800,8B71,0308,0470,1807 0043  7719 ;% 06    Q_R[TEMP2].RL.P                         ;ROTATE LOOP COUNT LEFT 2
U 07, 7700,4800,03A4,03D8,4870,1808 004E  7723 ;% 07    WCTRL/PC_WB,WB_Q                        ;FORCE PREFETCH TO HAPPEN
U 08, 7700,4800,0364,1700,0470,1809 0059  7727 ;% 08    IRD1TEST                                ;LOAD IR REGISTER
U 09, 7600,4800,0364,0300,0470,180A 0064  7731 ;% 09    DTYPE/IDEP,STROBE.V.BUS                 ;LOAD DSIZE INTO VBUS
U 0A, 7300,C800,0364,0300,0470,180B 006F  7735 ;% 0A    NOP,STOP.CLOCK                          ;STOP THE CPU CLOCK
U 0B, 7700,4800,0364,0300,0470,180C 007A  7739 ;% 0B    NOP
U 0C, 7700,C800,0364,1B00,0470,180D 0085  7743 ;% 0C    LOD INC BRA?                            ;INCREMENT THE IRD COUNTER
U 0D, 7600,C800,0364,0300,0470,180E 0090  7747 ;% 0D    DTYPE/IDEP,STROBE.V.BUS                 ;LOAD DSIZE INTO VBUS
U 0E, 7300,4800,0364,0300,0470,180F 009B  7751 ;% 0E    NOP,STOP.CLOCK                          ;STOP THE CPU CLOCK
                                 00A6  7755
                                 00A6  7756 RTEMP_DATA
                                 0001  7757
                          00C00000 0001  7758          .LONG   ^X00C00000                      ;DISABLE ^P INTERRUPT
                          00400000 0005  7759          .LONG   ^X00400000                      ;...
                                 0009  7760
                                 0009  7761 CACHE_DATA
                                 0001  7762
                          0000001A 0001  7763          .LONG   ^X0000001A                      ;BGTRU
                          0000001B 0005  7764          .LONG   ^X0000001B                      ;BLEQU
                          0000001C 0009  7765          .LONG   ^X0000001C                      ;BVC
                          0000001D 000D  7766          .LONG   ^X0000001D                      ;BVS
                          0000001E 0011  7767          .LONG   ^X0000001E                      ;BGEQU
```

```
00000001F  0015  7768          .LONG   ^X0000001F                      ;BLSSU
00000020   0019  7769          .LONG   ^X00000020                      ;ADDP4
00000021   001D  7770          .LONG   ^X00000021                      ;ADDP6
00000022   0021  7771          .LONG   ^X00000022                      ;SUBP4
00000023   0025  7772          .LONG   ^X00000023                      ;SUBP6
00000024   0029  7773          .LONG   ^X00000024                      ;CVTPT
00000025   002D  7774          .LONG   ^X00000025                      ;MULP
00000026   0031  7775          .LONG   ^X00000026                      ;CVTTP
00000027   0035  7776          .LONG   ^X00000027                      ;DIVP
00000028   0039  7777          .LONG   ^X00000028                      ;MOVC3
00000029   003D  7778          .LONG   ^X00000029                      ;CMPC3
0000002A   0041  7779          .LONG   ^X0000002A                      ;SCANC
0000002B   0045  7780          .LONG   ^X0000002B                      ;SPANC
0000002C   0049  7781          .LONG   ^X0000002C                      ;MOVC5
0000002D   004D  7782          .LONG   ^X0000002D                      ;CMPC5
0000002E   0051  7783          .LONG   ^X0000002E                      ;MOVTC
0000002F   0055  7784          .LONG   ^X0000002F                      ;MOVTUC
00000030   0059  7785          .LONG   ^X00000030                      ;BSBW
00000031   005D  7786          .LONG   ^X00000031                      ;BRW
00000032   0061  7787          .LONG   ^X00000032                      ;CVTWL
00000000   0065  7788          .LONG   ^X00000000
           0069  7789
           0069  7790 END_CPU_DATA
           0243  7791
           0243  7792
           0243  7793 END_TEST_DATA
           0243
           0243       ;-----------------------------------------------------------------------------
           0243  7794 ENDTEST
```

```
                        .SBTTL  "TEST   086     NATIVE MODE DSIZE ROM TEST 4
0021   7796 ;*******************************************************************************
0021   7797 ;++
0021   7798 ;
0021   7799 ; FUNCTIONAL DESCRIPTION:
0021   7800 ;
0021   7801 ;       THIS IS ONE IN A SERIES OF 14 TESTS THAT CHECK THE CONTENTS OF THE
0021   7802 ;       DSIZE ROM.
0021   7803 ;
0021   7804 ;
0021   7805 ; TEST ALGORITHM:
0021   7806 ;
0021   7807 ;       1.  CREATE AN INDEX (K) TO INDEX RESULTS TABLE
0021   7808 ;       2.  CREATE LOOP (I) OF 25
0021   7809 ;       3.  SAVE LOOP COUNT IN 1$
0021   7810 ;       4.  MOVE LOOP COUNT TO WDREG
0021   7811 ;       5.  EXECUTE DCS PROGRAM
0021   7812 ;               a.  DISABLE ^P INTERRUPTS
0021   7813 ;               b.  SET IPL TO 1F
0021   7814 ;               c.  MOVE LOOP COUNT TO RTEMP2
0021   7815 ;               d.  ROTATE LOOP COUNT LEFT 2 TO FORM CACHE ADDRESS
0021   7816 ;               e.  LOAD THIS ADDRESS INTO PC TO CAUSE PREFETCH
0021   7817 ;               f.  PERFORM IRD1TEST TO CAUSE IR TO LOAD INSTRUCTION
0021   7818 ;               g.  SET DTYPE TO INSTURCTION DEPENDENT AND STROBE THE VBUS
0021   7819 ;       6.  TEST VBUS FOR CORRECT DSIZE
0021   7820 ;       7.  IF NO ERROR CREATE LOOP (J) OF 5
0021   7821 ;       8.  EXECUTE DCS PROGRAM
0021   7822 ;               a.  INCREMENT THE IRD COUNTER
0021   7823 ;               b.  SET DTYPE TO INSTRUCTION DEPENDENT AND STROBE THE VBUS
0021   7824 ;       9.  TEST VBUS FOR CORRECT DSIZE
0021   7825 ;      10.  INCREMENT INDEX K BY ONE
0021   7826 ;      11.  IF NO ERROR GO TO STEP 8 UNTIL LOOP J IS EXHAUSTED
0021   7827 ;      12.  GO TO STEP 3 UNTIL LOOP I IS EXHAUSTED
0021   7828 ;
0021   7829 ;
0021   7830 ; TEST PATTERNS:
0021   7831 ;
0021   7832 ;       LOOP COUNT I            INSTRUCTION
0021   7833 ;            1                     CVTWB
0021   7834 ;            2                     MOVP
0021   7835 ;            3                     CMPP3
0021   7836 ;            4                     CVTPL
0021   7837 ;            5                     CMPP4
0021   7838 ;            6                     EDITPC
0021   7839 ;            7                     MATCHC
0021   7840 ;            8                     LOCC
0021   7841 ;            9                     SKPC
0021   7842 ;            A                     MOVZWL
0021   7843 ;            B                     ACBW
0021   7844 ;            C                     MOVAW
0021   7845 ;            D                     PUSHAW
0021   7846 ;            E                     ADDF2
0021   7847 ;            F                     ADDF3
0021   7848 ;           10                     SUBF2
0021   7849 ;           11                     SUBF3
```

```
                        0021  7850 ;           12                    MULF2
                        0021  7851 ;           13                    MULF3
                        0021  7852 ;           14                    DIVF2
                        0021  7853 ;           15                    DIVF3
                        0021  7854 ;           16                    CVTFB
                        0021  7855 ;           17                    CVTFW
                        0021  7856 ;           18                    CVTFL
                        0021  7857 ;           19                    CVTRFL
                        0021  7858 ;
                        0021  7859 ;
                        0021  7860 ; LOGIC DESCRIPTION:
                        0021  7861 ;
                        0021  7862 ;       THIS TEST ALONG WITH THE OTHER 13 TESTS IN THIS SERIES CHECK ALL THE
                        0021  7863 ;       LOCATIONS BLASTED INTO THE DSIZE ROM.  ANY FALIURES SHOULD BE CONFINED
                        0021  7864 ;       TO THE ROM.
                        0021  7865 ;
                        0021  7866 ;
                        0021  7867 ; ASSUMPTIONS:
                        0021  7868 ;
                        0021  7869 ;       THIS TEST ASSUMES THE PREFETCH LOGIC AND INSTRUCTION REGISTER ARE
                        0021  7870 ;       OPERATIONAL.
                        0021  7871 ;
                        0021  7872 ;
                        0021  7873 ; ERROR DESCRIPTION:
                        0021  7874 ;
                        0021  7875 ;       EXPECTED VBUS DATA (DSIZE)
                        0021  7876 ;       RECEIVED VBUS DATA
                        0021  7877 ;       LOOP COUNT I
                        0021  7878 ;       LOOP COUNT J
                        0021  7879 ;       LOOP COUNT K
                        0021  7880 ;               (BITS <7:0> = VBUS BIT POSITION)
                        0021  7881 ;               (BITS <8> = BIT STATE)
                        0021  7882 ;
                        0021  7883 ;--
                        0021  7884 ;*******************************************************************************
                        0021  7885
                        0021  7886
                        0021  7887          INITIALIZE                          ;INIT THE CPU
                        0023  7888          LDINDEX         K,1                 ;INIT INDEX K TO 1
                        002A  7889          LOOP            I,1,25              ;CREATE A TEST LOOP OF 50
                        0031  7890          SAVEINDEX       I,1$                ;SAVE LOOP COUNT IN 1$
                        0036  7891          LOADREG         WDREG,1$,,L         ;LOAD LOOP COUNT INTO WDREG
                        003E  7892          ERRLOOP                             ;ERROR LOOP POINT
                        0040  7893          FETCH           0                   ;START DCS PROGRAM
                        0045  7894          BRSTCLK         <^X0A>,INH          ;END DCS PROGRAM
                        0049  7895          CMPVBUS         2$,I                ;TEST DSIZE
                        004E  7896          IFERROR                             ;ROM FAILURE
                        0053  7897          LOOP            J,1,5               ;CREATE TEST LOOP OF 5
                        005A  7898          FETCH           <^X0B>              ;START DCS PROGRAM
                        005F  7899          BRSTCLK         <^X0E>,INH          ;END DCS PROGRAM
                        0063  7900          CMPVBUS         3$,K                ;TEST DSIZE
                        0068  7901          IFERROR                             ;ROM FAILURE
                        006D  7902          INCINDEX        K                   ;INCREMENT INDEX K BY ONE
                        0070  7903          ENDLOOP         J                   ;END TEST LOOP J
                        0073  7904          ENDLOOP         I                   ;END TEST LOOP I
```

```
                     0076  7905          SKIP                                ;GO TO NEXT TEST
                     007D  7906
                     007D  7907
                     007D  7908 BEGIN_TEST_DATA
                     007D
                     007D          ;-------------------------------------------------------------------------
                     007D  7909
                00000000  007D  7910 1$:      .LONG    ^X00000000                ;LOOP COUNT STORAGE
                      02  0081  7911 2$:      .BYTE    ^X2                       ;RESULTS TABLE (IRD 0)
                          0082  7912          VBUSDATA         <^X24>,1          ;CVTWB
                          0083  7913          VBUSDATA         <^X25>,0
                      02  0084  7914          .BYTE    ^X2                       ;MOVP
                          0085  7915          VBUSDATA         <^X24>,1
                          0086  7916          VBUSDATA         <^X25>,0
                      02  0087  7917          .BYTE    ^X2                       ;CMPP3
                          0088  7918          VBUSDATA         <^X24>,1
                          0089  7919          VBUSDATA         <^X25>,0
                      02  008A  7920          .BYTE    ^X2                       ;CVTPL
                          008B  7921          VBUSDATA         <^X24>,1
                          008C  7922          VBUSDATA         <^X25>,0
                      02  008D  7923          .BYTE    ^X2                       ;CMPP4
                          008E  7924          VBUSDATA         <^X24>,1
                          008F  7925          VBUSDATA         <^X25>,0
                      02  0090  7926          .BYTE    ^X2                       ;EDITPC
                          0091  7927          VBUSDATA         <^X24>,1
                          0092  7928          VBUSDATA         <^X25>,0
                      02  0093  7929          .BYTE    ^X2                       ;MATCHC
                          0094  7930          VBUSDATA         <^X24>,1
                          0095  7931          VBUSDATA         <^X25>,0
                      02  0096  7932          .BYTE    ^X2                       ;LOCC
                          0097  7933          VBUSDATA         <^X24>,0
                          0098  7934          VBUSDATA         <^X25>,0
                      02  0099  7935          .BYTE    ^X2                       ;SKPC
                          009A  7936          VBUSDATA         <^X24>,0
                          009B  7937          VBUSDATA         <^X25>,0
                      02  009C  7938          .BYTE    ^X2                       ;MOVZWL
                          009D  7939          VBUSDATA         <^X24>,1
                          009E  7940          VBUSDATA         <^X25>,0
                      02  009F  7941          .BYTE    ^X2                       ;ACBW
                          00A0  7942          VBUSDATA         <^X24>,1
                          00A1  7943          VBUSDATA         <^X25>,0
                      02  00A2  7944          .BYTE    ^X2                       ;MOVAW
                          00A3  7945          VBUSDATA         <^X24>,1
                          00A4  7946          VBUSDATA         <^X25>,0
                      02  00A5  7947          .BYTE    ^X2                       ;PUSHAW
                          00A6  7948          VBUSDATA         <^X24>,1
                          00A7  7949          VBUSDATA         <^X25>,0
                      02  00A8  7950          .BYTE    ^X2                       ;ADDF2
                          00A9  7951          VBUSDATA         <^X24>,0
                          00AA  7952          VBUSDATA         <^X25>,1
                      02  00AB  7953          .BYTE    ^X2                       ;ADDF3
                          00AC  7954          VBUSDATA         <^X24>,0
                          00AD  7955          VBUSDATA         <^X25>,1
                      02  00AE  7956          .BYTE    ^X2                       ;SUBF2
                          00AF  7957          VBUSDATA         <^X24>,0
```

```
                    00B0  7958          VBUSDATA        <^X25>,1
                02  00B1  7959          .BYTE   ^X2                             ;SUBF3
                    00B2  7960          VBUSDATA        <^X24>,0
                    00B3  7961          VBUSDATA        <^X25>,1
                02  00B4  7962          .BYTE   ^X2                             ;MULF2
                    00B5  7963          VBUSDATA        <^X24>,0
                    00B6  7964          VBUSDATA        <^X25>,1
                02  00B7  7965          .BYTE   ^X2                             ;MULF3
                    00B8  7966          VBUSDATA        <^X24>,0
                    00B9  7967          VBUSDATA        <^X25>,1
                02  00BA  7968          .BYTE   ^X2                             ;DIVF2
                    00BB  7969          VBUSDATA        <^X24>,0
                    00BC  7970          VBUSDATA        <^X25>,1
                02  00BD  7971          .BYTE   ^X2                             ;DIVF3
                    00BE  7972          VBUSDATA        <^X24>,0
                    00BF  7973          VBUSDATA        <^X25>,1
                02  00C0  7974          .BYTE   ^X2                             ;CVTFB
                    00C1  7975          VBUSDATA        <^X24>,0
                    00C2  7976          VBUSDATA        <^X25>,1
                02  00C3  7977          .BYTE   ^X2                             ;CVTFW
                    00C4  7978          VBUSDATA        <^X24>,0
                    00C5  7979          VBUSDATA        <^X25>,1
                02  00C6  7980          .BYTE   ^X2                             ;CVTFL
                    00C7  7981          VBUSDATA        <^X24>,0
                    00C8  7982          VBUSDATA        <^X25>,1
                02  00C9  7983          .BYTE   ^X2                             ;CVTRFL
                    00CA  7984          VBUSDATA        <^X24>,0
                    00CB  7985          VBUSDATA        <^X25>,1
                    00CC  7986
                02  00CC  7987 3$:      .BYTE   ^X2                             ;RESULTS TABLE (IRD 1 TO 5)
                    00CD  7988          VBUSDATA        <^X24>,0                ;CVTWB
                    00CE  7989          VBUSDATA        <^X25>,0
                02  00CF  7990          .BYTE   ^X2
                    00D0  7991          VBUSDATA        <^X24>,0
                    00D1  7992          VBUSDATA        <^X25>,0
                02  00D2  7993          .BYTE   ^X2
                    00D3  7994          VBUSDATA        <^X24>,0
                    00D4  7995          VBUSDATA        <^X25>,0
                02  00D5  7996          .BYTE   ^X2
                    00D6  7997          VBUSDATA        <^X24>,0
                    00D7  7998          VBUSDATA        <^X25>,0
                02  00D8  7999          .BYTE   ^X2
                    00D9  8000          VBUSDATA        <^X24>,0
                    00DA  8001          VBUSDATA        <^X25>,0
                02  00DB  8002          .BYTE   ^X2                             ;MOVP
                    00DC  8003          VBUSDATA        <^X24>,0
                    00DD  8004          VBUSDATA        <^X25>,0
                02  00DE  8005          .BYTE   ^X2
                    00DF  8006          VBUSDATA        <^X24>,0
                    00E0  8007          VBUSDATA        <^X25>,0
                02  00E1  8008          .BYTE   ^X2
                    00E2  8009          VBUSDATA        <^X24>,0
                    00E3  8010          VBUSDATA        <^X25>,0
                02  00E4  8011          .BYTE   ^X2
                    00E5  8012          VBUSDATA        <^X24>,0
```

```
                    00E6  8013          VBUSDATA        <^X25>,0
                02  00E7  8014          .BYTE   ^X2
                    00E8  8015          VBUSDATA        <^X24>,0
                    00E9  8016          VBUSDATA        <^X25>,0
                02  00EA  8017          .BYTE   ^X2                             ;CMPP3
                    00EB  8018          VBUSDATA        <^X24>,0
                    00EC  8019          VBUSDATA        <^X25>,0
                02  00ED  8020          .BYTE   ^X2
                    00EE  8021          VBUSDATA        <^X24>,0
                    00EF  8022          VBUSDATA        <^X25>,0
                02  00F0  8023          .BYTE   ^X2
                    00F1  8024          VBUSDATA        <^X24>,0
                    00F2  8025          VBUSDATA        <^X25>,0
                02  00F3  8026          .BYTE   ^X2
                    00F4  8027          VBUSDATA        <^X24>,0
                    00F5  8028          VBUSDATA        <^X25>,0
                02  00F6  8029          .BYTE   ^X2
                    00F7  8030          VBUSDATA        <^X24>,0
                    00F8  8031          VBUSDATA        <^X25>,0
                02  00F9  8032          .BYTE   ^X2                             ;CVTPL
                    00FA  8033          VBUSDATA        <^X24>,0
                    00FB  8034          VBUSDATA        <^X25>,0
                02  00FC  8035          .BYTE   ^X2
                    00FD  8036          VBUSDATA        <^X24>,0
                    00FE  8037          VBUSDATA        <^X25>,1
                02  00FF  8038          .BYTE   ^X2
                    0100  8039          VBUSDATA        <^X24>,0
                    0101  8040          VBUSDATA        <^X25>,0
                02  0102  8041          .BYTE   ^X2
                    0103  8042          VBUSDATA        <^X24>,0
                    0104  8043          VBUSDATA        <^X25>,0
                02  0105  8044          .BYTE   ^X2
                    0106  8045          VBUSDATA        <^X24>,0
                    0107  8046          VBUSDATA        <^X25>,0
                02  0108  8047          .BYTE   ^X2                             ;CMPP4
                    0109  8048          VBUSDATA        <^X24>,0
                    010A  8049          VBUSDATA        <^X25>,0
                02  010B  8050          .BYTE   ^X2
                    010C  8051          VBUSDATA        <^X24>,1
                    010D  8052          VBUSDATA        <^X25>,0
                02  010E  8053          .BYTE   ^X2
                    010F  8054          VBUSDATA        <^X24>,0
                    0110  8055          VBUSDATA        <^X25>,0
                02  0111  8056          .BYTE   ^X2
                    0112  8057          VBUSDATA        <^X24>,0
                    0113  8058          VBUSDATA        <^X25>,0
                02  0114  8059          .BYTE   ^X2
                    0115  8060          VBUSDATA        <^X24>,0
                    0116  8061          VBUSDATA        <^X25>,0
                02  0117  8062          .BYTE   ^X2                             ;EDITPC
                    0118  8063          VBUSDATA        <^X24>,0
                    0119  8064          VBUSDATA        <^X25>,0
                02  011A  8065          .BYTE   ^X2
                    011B  8066          VBUSDATA        <^X24>,0
                    011C  8067          VBUSDATA        <^X25>,0
```

```
                02  011D  8068          .BYTE   ^X2
                    011E  8069          VBUSDATA        <^X24>,0
                    011F  8070          VBUSDATA        <^X25>,0
                02  0120  8071          .BYTE   ^X2
                    0121  8072          VBUSDATA        <^X24>,0
                    0122  8073          VBUSDATA        <^X25>,0
                02  0123  8074          .BYTE   ^X2
                    0124  8075          VBUSDATA        <^X24>,0
                    0125  8076          VBUSDATA        <^X25>,0
                02  0126  8077          .BYTE   ^X2                             ;MATCHC
                    0127  8078          VBUSDATA        <^X24>,0
                    0128  8079          VBUSDATA        <^X25>,0
                02  0129  8080          .BYTE   ^X2
                    012A  8081          VBUSDATA        <^X24>,1
                    012B  8082          VBUSDATA        <^X25>,0
                02  012C  8083          .BYTE   ^X2
                    012D  8084          VBUSDATA        <^X24>,0
                    012E  8085          VBUSDATA        <^X25>,0
                02  012F  8086          .BYTE   ^X2
                    0130  8087          VBUSDATA        <^X24>,0
                    0131  8088          VBUSDATA        <^X25>,0
                02  0132  8089          .BYTE   ^X2
                    0133  8090          VBUSDATA        <^X24>,0
                    0134  8091          VBUSDATA        <^X25>,0
                02  0135  8092          .BYTE   ^X2                             ;LOCC
                    0136  8093          VBUSDATA        <^X24>,1
                    0137  8094          VBUSDATA        <^X25>,0
                02  0138  8095          .BYTE   ^X2
                    0139  8096          VBUSDATA        <^X24>,0
                    013A  8097          VBUSDATA        <^X25>,0
                02  013B  8098          .BYTE   ^X2
                    013C  8099          VBUSDATA        <^X24>,0
                    013D  8100          VBUSDATA        <^X25>,0
                02  013E  8101          .BYTE   ^X2
                    013F  8102          VBUSDATA        <^X24>,0
                    0140  8103          VBUSDATA        <^X25>,0
                02  0141  8104          .BYTE   ^X2
                    0142  8105          VBUSDATA        <^X24>,0
                    0143  8106          VBUSDATA        <^X25>,0
                02  0144  8107          .BYTE   ^X2                             ;SKPC
                    0145  8108          VBUSDATA        <^X24>,1
                    0146  8109          VBUSDATA        <^X25>,0
                02  0147  8110          .BYTE   ^X2
                    0148  8111          VBUSDATA        <^X24>,0
                    0149  8112          VBUSDATA        <^X25>,0
                02  014A  8113          .BYTE   ^X2
                    014B  8114          VBUSDATA        <^X24>,0
                    014C  8115          VBUSDATA        <^X25>,0
                02  014D  8116          .BYTE   ^X2
                    014E  8117          VBUSDATA        <^X24>,0
                    014F  8118          VBUSDATA        <^X25>,0
                02  0150  8119          .BYTE   ^X2
                    0151  8120          VBUSDATA        <^X24>,0
                    0152  8121          VBUSDATA        <^X25>,0
                02  0153  8122          .BYTE   ^X2                             ;MOVZWL
```

```
                    0154  8123          VBUSDATA        <^X24>,0
                    0155  8124          VBUSDATA        <^X25>,1
                02  0156  8125          .BYTE   ^X2
                    0157  8126          VBUSDATA        <^X24>,0
                    0158  8127          VBUSDATA        <^X25>,0
                02  0159  8128          .BYTE   ^X2
                    015A  8129          VBUSDATA        <^X24>,0
                    015B  8130          VBUSDATA        <^X25>,0
                02  015C  8131          .BYTE   ^X2
                    015D  8132          VBUSDATA        <^X24>,0
                    015E  8133          VBUSDATA        <^X25>,0
                02  015F  8134          .BYTE   ^X2
                    0160  8135          VBUSDATA        <^X24>,0
                    0161  8136          VBUSDATA        <^X25>,0
                02  0162  8137          .BYTE   ^X2                             ;ACBW
                    0163  8138          VBUSDATA        <^X24>,1
                    0164  8139          VBUSDATA        <^X25>,0
                02  0165  8140          .BYTE   ^X2
                    0166  8141          VBUSDATA        <^X24>,1
                    0167  8142          VBUSDATA        <^X25>,0
                02  0168  8143          .BYTE   ^X2
                    0169  8144          VBUSDATA        <^X24>,0
                    016A  8145          VBUSDATA        <^X25>,0
                02  016B  8146          .BYTE   ^X2
                    016C  8147          VBUSDATA        <^X24>,0
                    016D  8148          VBUSDATA        <^X25>,0
                02  016E  8149          .BYTE   ^X2
                    016F  8150          VBUSDATA        <^X24>,0
                    0170  8151          VBUSDATA        <^X25>,0
                02  0171  8152          .BYTE   ^X2                             ;MOVAW
                    0172  8153          VBUSDATA        <^X24>,0
                    0173  8154          VBUSDATA        <^X25>,1
                02  0174  8155          .BYTE   ^X2
                    0175  8156          VBUSDATA        <^X24>,0
                    0176  8157          VBUSDATA        <^X25>,0
                02  0177  8158          .BYTE   ^X2
                    0178  8159          VBUSDATA        <^X24>,0
                    0179  8160          VBUSDATA        <^X25>,0
                02  017A  8161          .BYTE   ^X2
                    017B  8162          VBUSDATA        <^X24>,0
                    017C  8163          VBUSDATA        <^X25>,0
                02  017D  8164          .BYTE   ^X2
                    017E  8165          VBUSDATA        <^X24>,0
                    017F  8166          VBUSDATA        <^X25>,0
                02  0180  8167          .BYTE   ^X2                             ;PUSHAW
                    0181  8168          VBUSDATA        <^X24>,0
                    0182  8169          VBUSDATA        <^X25>,0
                02  0183  8170          .BYTE   ^X2
                    0184  8171          VBUSDATA        <^X24>,0
                    0185  8172          VBUSDATA        <^X25>,0
                02  0186  8173          .BYTE   ^X2
                    0187  8174          VBUSDATA        <^X24>,0
                    0188  8175          VBUSDATA        <^X25>,0
                02  0189  8176          .BYTE   ^X2
                    018A  8177          VBUSDATA        <^X24>,0
```

```
                018B  8178          VBUSDATA        <^X25>,0
            02  018C  8179          .BYTE   ^X2
                018D  8180          VBUSDATA        <^X24>,0
                018E  8181          VBUSDATA        <^X25>,0
            02  018F  8182          .BYTE   ^X2                             ;ADDF2
                0190  8183          VBUSDATA        <^X24>,0
                0191  8184          VBUSDATA        <^X25>,1
            02  0192  8185          .BYTE   ^X2
                0193  8186          VBUSDATA        <^X24>,0
                0194  8187          VBUSDATA        <^X25>,0
            02  0195  8188          .BYTE   ^X2
                0196  8189          VBUSDATA        <^X24>,0
                0197  8190          VBUSDATA        <^X25>,0
            02  0198  8191          .BYTE   ^X2
                0199  8192          VBUSDATA        <^X24>,0
                019A  8193          VBUSDATA        <^X25>,0
            02  019B  8194          .BYTE   ^X2
                019C  8195          VBUSDATA        <^X24>,0
                019D  8196          VBUSDATA        <^X25>,0
            02  019E  8197          .BYTE   ^X2                             ;ADDF3
                019F  8198          VBUSDATA        <^X24>,0
                01A0  8199          VBUSDATA        <^X25>,1
            02  01A1  8200          .BYTE   ^X2
                01A2  8201          VBUSDATA        <^X24>,0
                01A3  8202          VBUSDATA        <^X25>,1
            02  01A4  8203          .BYTE   ^X2
                01A5  8204          VBUSDATA        <^X24>,0
                01A6  8205          VBUSDATA        <^X25>,0
            02  01A7  8206          .BYTE   ^X2
                01A8  8207          VBUSDATA        <^X24>,0
                01A9  8208          VBUSDATA        <^X25>,0
            02  01AA  8209          .BYTE   ^X2
                01AB  8210          VBUSDATA        <^X24>,0
                01AC  8211          VBUSDATA        <^X25>,0
            02  01AD  8212          .BYTE   ^X2                             ;SUBF2
                01AE  8213          VBUSDATA        <^X24>,0
                01AF  8214          VBUSDATA        <^X25>,1
            02  01B0  8215          .BYTE   ^X2
                01B1  8216          VBUSDATA        <^X24>,0
                01B2  8217          VBUSDATA        <^X25>,0
            02  01B3  8218          .BYTE   ^X2
                01B4  8219          VBUSDATA        <^X24>,0
                01B5  8220          VBUSDATA        <^X25>,0
            02  01B6  8221          .BYTE   ^X2
                01B7  8222          VBUSDATA        <^X24>,0
                01B8  8223          VBUSDATA        <^X25>,0
            02  01B9  8224          .BYTE   ^X2
                01BA  8225          VBUSDATA        <^X24>,0
                01BB  8226          VBUSDATA        <^X25>,0
            02  01BC  8227          .BYTE   ^X2                             ;SUBF3
                01BD  8228          VBUSDATA        <^X24>,0
                01BE  8229          VBUSDATA        <^X25>,1
            02  01BF  8230          .BYTE   ^X2
                01C0  8231          VBUSDATA        <^X24>,0
                01C1  8232          VBUSDATA        <^X25>,1
```

```
                    02  01C2  8233          .BYTE   ^X2
                        01C3  8234          VBUSDATA        <^X24>,0
                        01C4  8235          VBUSDATA        <^X25>,0
                    02  01C5  8236          .BYTE   ^X2
                        01C6  8237          VBUSDATA        <^X24>,0
                        01C7  8238          VBUSDATA        <^X25>,0
                    02  01C8  8239          .BYTE   ^X2
                        01C9  8240          VBUSDATA        <^X24>,0
                        01CA  8241          VBUSDATA        <^X25>,0
                    02  01CB  8242          .BYTE   ^X2                             ;MULF2
                        01CC  8243          VBUSDATA        <^X24>,0
                        01CD  8244          VBUSDATA        <^X25>,1
                    02  01CE  8245          .BYTE   ^X2
                        01CF  8246          VBUSDATA        <^X24>,0
                        01D0  8247          VBUSDATA        <^X25>,0
                    02  01D1  8248          .BYTE   ^X2
                        01D2  8249          VBUSDATA        <^X24>,0
                        01D3  8250          VBUSDATA        <^X25>,0
                    02  01D4  8251          .BYTE   ^X2
                        01D5  8252          VBUSDATA        <^X24>,0
                        01D6  8253          VBUSDATA        <^X25>,0
                    02  01D7  8254          .BYTE   ^X2
                        01D8  8255          VBUSDATA        <^X24>,0
                        01D9  8256          VBUSDATA        <^X25>,0
                    02  01DA  8257          .BYTE   ^X2                             ;MULF3
                        01DB  8258          VBUSDATA        <^X24>,0
                        01DC  8259          VBUSDATA        <^X25>,1
                    02  01DD  8260          .BYTE   ^X2
                        01DE  8261          VBUSDATA        <^X24>,0
                        01DF  8262          VBUSDATA        <^X25>,1
                    02  01E0  8263          .BYTE   ^X2
                        01E1  8264          VBUSDATA        <^X24>,0
                        01E2  8265          VBUSDATA        <^X25>,0
                    02  01E3  8266          .BYTE   ^X2
                        01E4  8267          VBUSDATA        <^X24>,0
                        01E5  8268          VBUSDATA        <^X25>,0
                    02  01E6  8269          .BYTE   ^X2
                        01E7  8270          VBUSDATA        <^X24>,0
                        01E8  8271          VBUSDATA        <^X25>,0
                    02  01E9  8272          .BYTE   ^X2                             ;DIVF2
                        01EA  8273          VBUSDATA        <^X24>,0
                        01EB  8274          VBUSDATA        <^X25>,1
                    02  01EC  8275          .BYTE   ^X2
                        01ED  8276          VBUSDATA        <^X24>,0
                        01EE  8277          VBUSDATA        <^X25>,0
                    02  01EF  8278          .BYTE   ^X2
                        01F0  8279          VBUSDATA        <^X24>,0
                        01F1  8280          VBUSDATA        <^X25>,0
                    02  01F2  8281          .BYTE   ^X2
                        01F3  8282          VBUSDATA        <^X24>,0
                        01F4  8283          VBUSDATA        <^X25>,0
                    02  01F5  8284          .BYTE   ^X2
                        01F6  8285          VBUSDATA        <^X24>,0
                        01F7  8286          VBUSDATA        <^X25>,0
                    02  01F8  8287          .BYTE   ^X2                             ;DIVF3
```

```
                    01F9  8288          VBUSDATA        <^X24>,0
                    01FA  8289          VBUSDATA        <^X25>,1
                02  01FB  8290          .BYTE   ^X2
                    01FC  8291          VBUSDATA        <^X24>,0
                    01FD  8292          VBUSDATA        <^X25>,1
                02  01FE  8293          .BYTE   ^X2
                    01FF  8294          VBUSDATA        <^X24>,0
                    0200  8295          VBUSDATA        <^X25>,0
                02  0201  8296          .BYTE   ^X2
                    0202  8297          VBUSDATA        <^X24>,0
                    0203  8298          VBUSDATA        <^X25>,0
                02  0204  8299          .BYTE   ^X2
                    0205  8300          VBUSDATA        <^X24>,0
                    0206  8301          VBUSDATA        <^X25>,0
                02  0207  8302          .BYTE   ^X2                             ;CVTFB
                    0208  8303          VBUSDATA        <^X24>,0
                    0209  8304          VBUSDATA        <^X25>,0
                02  020A  8305          .BYTE   ^X2
                    020B  8306          VBUSDATA        <^X24>,0
                    020C  8307          VBUSDATA        <^X25>,0
                02  020D  8308          .BYTE   ^X2
                    020E  8309          VBUSDATA        <^X24>,0
                    020F  8310          VBUSDATA        <^X25>,0
                02  0210  8311          .BYTE   ^X2
                    0211  8312          VBUSDATA        <^X24>,0
                    0212  8313          VBUSDATA        <^X25>,0
                02  0213  8314          .BYTE   ^X2
                    0214  8315          VBUSDATA        <^X24>,0
                    0215  8316          VBUSDATA        <^X25>,0
                02  0216  8317          .BYTE   ^X2                             ;CVTFW
                    0217  8318          VBUSDATA        <^X24>,1
                    0218  8319          VBUSDATA        <^X25>,0
                02  0219  8320          .BYTE   ^X2
                    021A  8321          VBUSDATA        <^X24>,0
                    021B  8322          VBUSDATA        <^X25>,0
                02  021C  8323          .BYTE   ^X2
                    021D  8324          VBUSDATA        <^X24>,0
                    021E  8325          VBUSDATA        <^X25>,0
                02  021F  8326          .BYTE   ^X2
                    0220  8327          VBUSDATA        <^X24>,0
                    0221  8328          VBUSDATA        <^X25>,0
                02  0222  8329          .BYTE   ^X2
                    0223  8330          VBUSDATA        <^X24>,0
                    0224  8331          VBUSDATA        <^X25>,0
                02  0225  8332          .BYTE   ^X2                             ;CVTFL
                    0226  8333          VBUSDATA        <^X24>,0
                    0227  8334          VBUSDATA        <^X25>,1
                02  0228  8335          .BYTE   ^X2
                    0229  8336          VBUSDATA        <^X24>,0
                    022A  8337          VBUSDATA        <^X25>,0
                02  022B  8338          .BYTE   ^X2
                    022C  8339          VBUSDATA        <^X24>,0
                    022D  8340          VBUSDATA        <^X25>,0
                02  022E  8341          .BYTE   ^X2
                :   022F  8342          VBUSDATA        <^X24>,0
```

```
                                  0230  8343          VBUSDATA        <^X25>,0
                              02  0231  8344          .BYTE   ^X2
                                  0232  8345          VBUSDATA        <^X24>,0
                                  0233  8346          VBUSDATA        <^X25>,0
                              02  0234  8347          .BYTE   ^X2                                  ;CVTRFL
                                  0235  8348          VBUSDATA        <^X24>,0
                                  0236  8349          VBUSDATA        <^X25>,1
                              02  0237  8350          .BYTE   ^X2
                                  0238  8351          VBUSDATA        <^X24>,0
                                  0239  8352          VBUSDATA        <^X25>,0
                              02  023A  8353          .BYTE   ^X2
                                  023B  8354          VBUSDATA        <^X24>,0
                                  023C  8355          VBUSDATA        <^X25>,0
                              02  023D  8356          .BYTE   ^X2
                                  023E  8357          VBUSDATA        <^X24>,0
                                  023F  8358          VBUSDATA        <^X25>,0
                              02  0240  8359          .BYTE   ^X2
                                  0241  8360          VBUSDATA        <^X24>,0
                                  0242  8361          VBUSDATA        <^X25>,0
                                  0243  8362
                                  0243  8363
                                  0243  8364
                                  0243  8365 BEGIN_CPU_DATA
                                  0002  8366
                                  0002  8367 DCS_DATA
                                  0001  8368
U 00, 7700,C800,0364,0300,0470,1801  0001  8369 ;% 00   NOP
U 01, 7700,4800,5BE4,0300,2470,1802  000C  8373 ;% 01   WCTRL/LOADCRAR,WB_R[TEMP0]                ;DISABLE ^P INTERRUPT
U 02, 7700,0800,5BE4,0304,2C70,1803  0017  8377 ;% 02   WCTRL/CONWRITE,WB_R[TEMP1]                ;...
U 03, 7700,9800,D370,030F,FA70,1804  0022  8381 ;% 03   IPL_[1F]                                  ;SET IPL TO 1F
U 04, 5700,4840,0364,0308,0470,1805  002D  8385 ;% 04   R[TEMP2]_RDM                              ;MOVE LOOP COUNT TO RTEMP2
U 05, 7700,D800,EF64,0301,0470,1806  0038  8389 ;% 05   PL_[2]                                    ;SET PLATCH TO 2
U 06, 7700,C800,8B71,0308,0470,1807  0043  8393 ;% 06   Q_R[TEMP2].RL.P                           ;ROTATE LOOP COUNT LEFT 2
U 07, 7700,4800,03A4,03D8,4870,1808  004E  8397 ;% 07   WCTRL/PC_WB,WB_Q                          ;FORCE PREFETCH TO HAPPEN
U 08, 7700,4800,0364,1700,0470,1809  0059  8401 ;% 08   IRD1TEST                                  ;LOAD IR REGISTER
U 09, 7600,4800,0364,0300,0470,180A  0064  8405 ;% 09   DTYPE/IDEP,STROBE.V.BUS                   ;LOAD DSIZE INTO VBUS
U 0A, 7300,C800,0364,0300,0470,180B  006F  8409 ;% 0A   NOP,STOP.CLOCK                            ;STOP THE CPU CLOCK
U 0B, 7700,4800,0364,0300,0470,180C  007A  8413 ;% 0B   NOP
U 0C, 7700,C800,0364,1B00,0470,180D  0085  8417 ;% 0C   LOD INC BRA?                              ;INCREMENT THE IRD COUNTER
U 0D, 7600,C800,0364,0300,0470,180E  0090  8421 ;% 0D   DTYPE/IDEP,STROBE.V.BUS                   ;LOAD DSIZE INTO VBUS
U 0E, 7300,4800,0364,0300,0470,180F  009B  8425 ;% 0E   NOP,STOP.CLOCK                            ;STOP THE CPU CLOCK
                                  00A6  8429
                                  00A6  8430 RTEMP_DATA
                                  0001  8431
                          00C00000  0001  8432          .LONG   ^X00C00000                        ;DISABLE ^P INTERRUPT
                          00400000  0005  8433          .LONG   ^X00400000                        ;...
                                  0009  8434
                                  0009  8435 CACHE_DATA
                                  0001  8436
                          00000033  0001  8437          .LONG   ^X00000033                        ;CVTWB
                          00000034  0005  8438          .LONG   ^X00000034                        ;MOVP
                          00000035  0009  8439          .LONG   ^X00000035                        ;CMPP3
                          00000036  000D  8440          .LONG   ^X00000036                        ;CVTPL
                          00000037  0011  8441          .LONG   ^X00000037                        ;CMPP4
                          00000038  0015  8442          .LONG   ^X00000038                        ;EDITPC
```

```
00000039  0019  8443          .LONG   ^X00000039                      ;MATCHC
0000003A  001D  8444          .LONG   ^X0000003A                      ;LOCC
0000003B  0021  8445          .LONG   ^X0000003B                      ;SKPC
0000003C  0025  8446          .LONG   ^X0000003C                      ;MOVZWL
0000003D  0029  8447          .LONG   ^X0000003D                      ;ACBW
0000003E  002D  8448          .LONG   ^X0000003E                      ;MOVAW
0000003F  0031  8449          .LONG   ^X0000003F                      ;PUSHAW
00000040  0035  8450          .LONG   ^X00000040                      ;ADDF2
00000041  0039  8451          .LONG   ^X00000041                      ;ADDF3
00000042  003D  8452          .LONG   ^X00000042                      ;SUBF2
00000043  0041  8453          .LONG   ^X00000043                      ;SUBF3
00000044  0045  8454          .LONG   ^X00000044                      ;MULF2
00000045  0049  8455          .LONG   ^X00000045                      ;MULF3
00000046  004D  8456          .LONG   ^X00000046                      ;DIVF2
00000047  0051  8457          .LONG   ^X00000047                      ;DIVF3
00000048  0055  8458          .LONG   ^X00000048                      ;CVTFB
00000049  0059  8459          .LONG   ^X00000049                      ;CVTFW
0000004A  005D  8460          .LONG   ^X0000004A                      ;CVTFL
0000004B  0061  8461          .LONG   ^X0000004B                      ;CVTRFL
00000000  0065  8462          .LONG   ^X00000000
          0069  8463
          0069  8464 END_CPU_DATA
          0243  8465
          0243  8466
          0243  8467 END_TEST_DATA
          0243
          0243      ;-----------------------------------------------------------------------------
          0243  8468 ENDTEST
```

```
                     0021        .SBTTL  "TEST   087     NATIVE MODE DSIZE ROM TEST 5
                     0021  8470 ;*******************************************************************************
                     0021  8471 ;++
                     0021  8472 ;
                     0021  8473 ; FUNCTIONAL DESCRIPTION:
                     0021  8474 ;
                     0021  8475 ;       THIS IS ONE IN A SERIES OF 14 TESTS THAT CHECK THE CONTENTS OF THE
                     0021  8476 ;       DSIZE ROM.
                     0021  8477 ;
                     0021  8478 ;
                     0021  8479 ; TEST ALGORITHM:
                     0021  8480 ;
                     0021  8481 ;       1.  CREATE AN INDEX (K) TO INDEX RESULTS TABLE
                     0021  8482 ;       2.  CREATE LOOP (I) OF 25
                     0021  8483 ;       3.  SAVE LOOP COUNT IN 1$
                     0021  8484 ;       4.  MOVE LOOP COUNT TO WDREG
                     0021  8485 ;       5.  EXECUTE DCS PROGRAM
                     0021  8486 ;               a.  DISABLE ^P INTERRUPTS
                     0021  8487 ;               b.  SET IPL TO 1F
                     0021  8488 ;               c.  MOVE LOOP COUNT TO RTEMP2
                     0021  8489 ;               d.  ROTATE LOOP COUNT LEFT 2 TO FORM CACHE ADDRESS
                     0021  8490 ;               e.  LOAD THIS ADDRESS INTO PC TO CAUSE PREFETCH
                     0021  8491 ;               f.  PERFORM IRD1TEST TO CAUSE IR TO LOAD INSTRUCTION
                     0021  8492 ;               g.  SET DTYPE TO INSTURCTION DEPENDENT AND STROBE THE VBUS
                     0021  8493 ;       6.  TEST VBUS FOR CORRECT DSIZE
                     0021  8494 ;       7.  IF NO ERROR CREATE LOOP (J) OF 5
                     0021  8495 ;       8.  EXECUTE DCS PROGRAM
                     0021  8496 ;               a.  INCREMENT THE IRD COUNTER
                     0021  8497 ;               b.  SET DTYPE TO INSTRUCTION DEPENDENT AND STROBE THE VBUS
                     0021  8498 ;       9.  TEST VBUS FOR CORRECT DSIZE
                     0021  8499 ;      10.  INCREMENT INDEX K BY ONE
                     0021  8500 ;      11.  IF NO ERROR GO TO STEP 8 UNTIL LOOP J IS EXHAUSTED
                     0021  8501 ;      12.  GO TO STEP 3 UNTIL LOOP I IS EXHAUSTED
                     0021  8502 ;
                     0021  8503 ;
                     0021  8504 ; TEST PATTERNS:
                     0021  8505 ;
                     0021  8506 ;       LOOP COUNT I            INSTRUCTION
                     0021  8507 ;            1                     CVTBF
                     0021  8508 ;            2                     CVTWF
                     0021  8509 ;            3                     CVTLF
                     0021  8510 ;            4                     ACBF
                     0021  8511 ;            5                     MOVF
                     0021  8512 ;            6                     CMPF
                     0021  8513 ;            7                     MNEGF
                     0021  8514 ;            8                     TSTF
                     0021  8515 ;            9                     EMODF
                     0021  8516 ;            A                     POLYF
                     0021  8517 ;            B                     CVTFD
                     0021  8518 ;            C                     RESVRD
                     0021  8519 ;            D                     ADAWI
                     0021  8520 ;            E                     RESVRD
                     0021  8521 ;            F                     RESVRD
                     0021  8522 ;           10                     RESVRD
                     0021  8523 ;           11                     INSQHI
```

```
0021  8524 ;           12                      INSQTI
0021  8525 ;           13                      REMQHI
0021  8526 ;           14                      REMQTI
0021  8527 ;           15                      ADDD2
0021  8528 ;           16                      ADDD3
0021  8529 ;           17                      SUBD2
0021  8530 ;           18                      SUBD3
0021  8531 ;           19                      MULD2
0021  8532 ;
0021  8533 ;
0021  8534 ; LOGIC DESCRIPTION:
0021  8535 ;
0021  8536 ;       THIS TEST ALONG WITH THE OTHER 13 TESTS IN THIS SERIES CHECK ALL THE
0021  8537 ;       LOCATIONS BLASTED INTO THE DSIZE ROM.  ANY FALIURES SHOULD BE CONFINED
0021  8538 ;       TO THE ROM.
0021  8539 ;
0021  8540 ;
0021  8541 ; ASSUMPTIONS:
0021  8542 ;
0021  8543 ;       THIS TEST ASSUMES THE PREFETCH LOGIC AND INSTRUCTION REGISTER ARE
0021  8544 ;       OPERATIONAL.
0021  8545 ;
0021  8546 ;
0021  8547 ; ERROR DESCRIPTION:
0021  8548 ;
0021  8549 ;       EXPECTED VBUS DATA (DSIZE)
0021  8550 ;       RECEIVED VBUS DATA
0021  8551 ;       LOOP COUNT I
0021  8552 ;       LOOP COUNT J
0021  8553 ;       LOOP COUNT K
0021  8554 ;               (BITS <7:0> = VBUS BIT POSITION)
0021  8555 ;               (BITS <8> = BIT STATE)
0021  8556 ;
0021  8557 ;--
0021  8558 ;*******************************************************************************
0021  8559
0021  8560
0021  8561         INITIALIZE                                      ;INIT THE CPU
0023  8562         LDINDEX         K,1                             ;INIT INDEX K TO 1
002A  8563         LOOP            I,1,25                          ;CREATE A TEST LOOP OF 50
0031  8564         SAVEINDEX       I,1$                            ;SAVE LOOP COUNT IN 1$
0036  8565         LOADREG         WDREG,1$,,L                     ;LOAD LOOP COUNT INTO WDREG
003E  8566         ERRLOOP                                         ;ERROR LOOP POINT
0040  8567         FETCH           0                               ;START DCS PROGRAM
0045  8568         BRSTCLK         <^X0A>,INH                      ;END DCS PROGRAM
0049  8569         CMPVBUS         2$,I                            ;TEST DSIZE
004E  8570         IFERROR                                         ;ROM FAILURE
0053  8571         LOOP            J,1,5                           ;CREATE TEST LOOP OF 5
005A  8572         FETCH           <^X0B>                          ;START DCS PROGRAM
005F  8573         BRSTCLK         <^X0E>,INH                      ;END DCS PROGRAM
0063  8574         CMPVBUS         3$,K                            ;TEST DSIZE
0068  8575         IFERROR                                         ;ROM FAILURE
006D  8576         INCINDEX        K                               ;INCREMENT INDEX K BY ONE
0070  8577         ENDLOOP         J                               ;END TEST LOOP J
0073  8578         ENDLOOP         I                               ;END TEST LOOP I
```

```
                    0076  8579          SKIP                                    ;GO TO NEXT TEST
                    007D  8580
                    007D  8581
                    007D  8582 BEGIN_TEST_DATA
                    007D
                    007D          ;-----------------------------------------------------------------------
                    007D  8583
          00000000  007D  8584 1$:      .LONG   ^X00000000                      ;LOOP COUNT STORAGE
                 02 0081  8585 2$:      .BYTE   ^X2                             ;RESULTS TABLE (IRD 0)
                    0082  8586          VBUSDATA        <^X24>,0                ;CVTBF
                    0083  8587          VBUSDATA        <^X25>,0
                 02 0084  8588          .BYTE   ^X2                             ;CVTWF
                    0085  8589          VBUSDATA        <^X24>,1
                    0086  8590          VBUSDATA        <^X25>,0
                 02 0087  8591          .BYTE   ^X2                             ;CVTLF
                    0088  8592          VBUSDATA        <^X24>,0
                    0089  8593          VBUSDATA        <^X25>,1
                 02 008A  8594          .BYTE   ^X2                             ;ACBF
                    008B  8595          VBUSDATA        <^X24>,0
                    008C  8596          VBUSDATA        <^X25>,1
                 02 008D  8597          .BYTE   ^X2                             ;MOVF
                    008E  8598          VBUSDATA        <^X24>,0
                    008F  8599          VBUSDATA        <^X25>,1
                 02 0090  8600          .BYTE   ^X2                             ;CMPF
                    0091  8601          VBUSDATA        <^X24>,0
                    0092  8602          VBUSDATA        <^X25>,1
                 02 0093  8603          .BYTE   ^X2                             ;MNEGF
                    0094  8604          VBUSDATA        <^X24>,0
                    0095  8605          VBUSDATA        <^X25>,1
                 02 0096  8606          .BYTE   ^X2                             ;TSTF
                    0097  8607          VBUSDATA        <^X24>,0
                    0098  8608          VBUSDATA        <^X25>,1
                 02 0099  8609          .BYTE   ^X2                             ;EMODF
                    009A  8610          VBUSDATA        <^X24>,0
                    009B  8611          VBUSDATA        <^X25>,1
                 02 009C  8612          .BYTE   ^X2                             ;POLYF
                    009D  8613          VBUSDATA        <^X24>,0
                    009E  8614          VBUSDATA        <^X25>,1
                 02 009F  8615          .BYTE   ^X2                             ;CVTFD
                    00A0  8616          VBUSDATA        <^X24>,0
                    00A1  8617          VBUSDATA        <^X25>,1
                 02 00A2  8618          .BYTE   ^X2                             ;RESVRD
                    00A3  8619          VBUSDATA        <^X24>,0
                    00A4  8620          VBUSDATA        <^X25>,0
                 02 00A5  8621          .BYTE   ^X2                             ;ADAWI
                    00A6  8622          VBUSDATA        <^X24>,1
                    00A7  8623          VBUSDATA        <^X25>,0
                 02 00A8  8624          .BYTE   ^X2                             ;RESVRD
                    00A9  8625          VBUSDATA        <^X24>,0
                    00AA  8626          VBUSDATA        <^X25>,0
                 02 00AB  8627          .BYTE   ^X2                             ;RESVRD
                    00AC  8628          VBUSDATA        <^X24>,0
                    00AD  8629          VBUSDATA        <^X25>,0
                 02 00AE  8630          .BYTE   ^X2                             ;RESVRD
                    00AF  8631          VBUSDATA        <^X24>,0
```

```
                    00B0  8632          VBUSDATA        <^X25>,0
02                  00B1  8633          .BYTE   ^X2                             ;INSQHI
                    00B2  8634          VBUSDATA        <^X24>,0
                    00B3  8635          VBUSDATA        <^X25>,0
02                  00B4  8636          .BYTE   ^X2                             ;INSQTI
                    00B5  8637          VBUSDATA        <^X24>,0
                    00B6  8638          VBUSDATA        <^X25>,0
02                  00B7  8639          .BYTE   ^X2                             ;REMQHI
                    00B8  8640          VBUSDATA        <^X24>,1
                    00B9  8641          VBUSDATA        <^X25>,1
02                  00BA  8642          .BYTE   ^X2                             ;REMQTI
                    00BB  8643          VBUSDATA        <^X24>,1
                    00BC  8644          VBUSDATA        <^X25>,1
02                  00BD  8645          .BYTE   ^X2                             ;ADDD2
                    00BE  8646          VBUSDATA        <^X24>,1
                    00BF  8647          VBUSDATA        <^X25>,1
02                  00C0  8648          .BYTE   ^X2                             ;ADDD3
                    00C1  8649          VBUSDATA        <^X24>,1
                    00C2  8650          VBUSDATA        <^X25>,1
02                  00C3  8651          .BYTE   ^X2                             ;SUBD2
                    00C4  8652          VBUSDATA        <^X24>,1
                    00C5  8653          VBUSDATA        <^X25>,1
02                  00C6  8654          .BYTE   ^X2                             ;SUBD3
                    00C7  8655          VBUSDATA        <^X24>,1
                    00C8  8656          VBUSDATA        <^X25>,1
02                  00C9  8657          .BYTE   ^X2                             ;MULD2
                    00CA  8658          VBUSDATA        <^X24>,1
                    00CB  8659          VBUSDATA        <^X25>,1
                    00CC  8660
02                  00CC  8661 3$:      .BYTE   ^X2                             ;RESULTS TABLE (IRD 1 TO 5)
                    00CD  8662          VBUSDATA        <^X24>,0                ;CVTBF
                    00CE  8663          VBUSDATA        <^X25>,1
02                  00CF  8664          .BYTE   ^X2
                    00D0  8665          VBUSDATA        <^X24>,0
                    00D1  8666          VBUSDATA        <^X25>,0
02                  00D2  8667          .BYTE   ^X2
                    00D3  8668          VBUSDATA        <^X24>,0
                    00D4  8669          VBUSDATA        <^X25>,0
02                  00D5  8670          .BYTE   ^X2
                    00D6  8671          VBUSDATA        <^X24>,0
                    00D7  8672          VBUSDATA        <^X25>,0
02                  00D8  8673          .BYTE   ^X2
                    00D9  8674          VBUSDATA        <^X24>,0
                    00DA  8675          VBUSDATA        <^X25>,0
02                  00DB  8676          .BYTE   ^X2                             ;CVTWF
                    00DC  8677          VBUSDATA        <^X24>,0
                    00DD  8678          VBUSDATA        <^X25>,1
02                  00DE  8679          .BYTE   ^X2
                    00DF  8680          VBUSDATA        <^X24>,0
                    00E0  8681          VBUSDATA        <^X25>,0
02                  00E1  8682          .BYTE   ^X2
                    00E2  8683          VBUSDATA        <^X24>,0
                    00E3  8684          VBUSDATA        <^X25>,0
02                  00E4  8685          .BYTE   ^X2
                    00E5  8686          VBUSDATA        <^X24>,0
```

```
                    00E6  8687          VBUSDATA        <^X25>,0
02                  00E7  8688          .BYTE   ^X2
                    00E8  8689          VBUSDATA        <^X24>,0
                    00E9  8690          VBUSDATA        <^X25>,0
02                  00EA  8691          .BYTE   ^X2                             ;CVTLF
                    00EB  8692          VBUSDATA        <^X24>,0
                    00EC  8693          VBUSDATA        <^X25>,1
02                  00ED  8694          .BYTE   ^X2
                    00EE  8695          VBUSDATA        <^X24>,0
                    00EF  8696          VBUSDATA        <^X25>,0
02                  00F0  8697          .BYTE   ^X2
                    00F1  8698          VBUSDATA        <^X24>,0
                    00F2  8699          VBUSDATA        <^X25>,0
02                  00F3  8700          .BYTE   ^X2
                    00F4  8701          VBUSDATA        <^X24>,0
                    00F5  8702          VBUSDATA        <^X25>,0
02                  00F6  8703          .BYTE   ^X2
                    00F7  8704          VBUSDATA        <^X24>,0
                    00F8  8705          VBUSDATA        <^X25>,0
02                  00F9  8706          .BYTE   ^X2                             ;ACBF
                    00FA  8707          VBUSDATA        <^X24>,0
                    00FB  8708          VBUSDATA        <^X25>,1
02                  00FC  8709          .BYTE   ^X2
                    00FD  8710          VBUSDATA        <^X24>,0
                    00FE  8711          VBUSDATA        <^X25>,1
02                  00FF  8712          .BYTE   ^X2
                    0100  8713          VBUSDATA        <^X24>,0
                    0101  8714          VBUSDATA        <^X25>,0
02                  0102  8715          .BYTE   ^X2
                    0103  8716          VBUSDATA        <^X24>,0
                    0104  8717          VBUSDATA        <^X25>,0
02                  0105  8718          .BYTE   ^X2
                    0106  8719          VBUSDATA        <^X24>,0
                    0107  8720          VBUSDATA        <^X25>,0
02                  0108  8721          .BYTE   ^X2                             ;MOVF
                    0109  8722          VBUSDATA        <^X24>,0
                    010A  8723          VBUSDATA        <^X25>,1
02                  010B  8724          .BYTE   ^X2
                    010C  8725          VBUSDATA        <^X24>,0
                    010D  8726          VBUSDATA        <^X25>,0
02                  010E  8727          .BYTE   ^X2
                    010F  8728          VBUSDATA        <^X24>,0
                    0110  8729          VBUSDATA        <^X25>,0
02                  0111  8730          .BYTE   ^X2
                    0112  8731          VBUSDATA        <^X24>,0
                    0113  8732          VBUSDATA        <^X25>,0
02                  0114  8733          .BYTE   ^X2
                    0115  8734          VBUSDATA        <^X24>,0
                    0116  8735          VBUSDATA        <^X25>,0
02                  0117  8736          .BYTE   ^X2                             ;CMPF
                    0118  8737          VBUSDATA        <^X24>,0
                    0119  8738          VBUSDATA        <^X25>,1
02                  011A  8739          .BYTE   ^X2
                    011B  8740          VBUSDATA        <^X24>,0
                    011C  8741          VBUSDATA        <^X25>,0
```

```
                    02  011D  8742          .BYTE   ^X2
                        011E  8743          VBUSDATA        <^X24>,0
                        011F  8744          VBUSDATA        <^X25>,0
                    02  0120  8745          .BYTE   ^X2
                        0121  8746          VBUSDATA        <^X24>,0
                        0122  8747          VBUSDATA        <^X25>,0
                    02  0123  8748          .BYTE   ^X2
                        0124  8749          VBUSDATA        <^X24>,0
                        0125  8750          VBUSDATA        <^X25>,0
                    02  0126  8751          .BYTE   ^X2                             ;MNEGF
                        0127  8752          VBUSDATA        <^X24>,0
                        0128  8753          VBUSDATA        <^X25>,1
                    02  0129  8754          .BYTE   ^X2
                        012A  8755          VBUSDATA        <^X24>,0
                        012B  8756          VBUSDATA        <^X25>,0
                    02  012C  8757          .BYTE   ^X2
                        012D  8758          VBUSDATA        <^X24>,0
                        012E  8759          VBUSDATA        <^X25>,0
                    02  012F  8760          .BYTE   ^X2
                        0130  8761          VBUSDATA        <^X24>,0
                        0131  8762          VBUSDATA        <^X25>,0
                    02  0132  8763          .BYTE   ^X2
                        0133  8764          VBUSDATA        <^X24>,0
                        0134  8765          VBUSDATA        <^X25>,0
                    02  0135  8766          .BYTE   ^X2                             ;TSTF
                        0136  8767          VBUSDATA        <^X24>,0
                        0137  8768          VBUSDATA        <^X25>,0
                    02  0138  8769          .BYTE   ^X2
                        0139  8770          VBUSDATA        <^X24>,0
                        013A  8771          VBUSDATA        <^X25>,0
                    02  013B  8772          .BYTE   ^X2
                        013C  8773          VBUSDATA        <^X24>,0
                        013D  8774          VBUSDATA        <^X25>,0
                    02  013E  8775          .BYTE   ^X2
                        013F  8776          VBUSDATA        <^X24>,0
                        0140  8777          VBUSDATA        <^X25>,0
                    02  0141  8778          .BYTE   ^X2
                        0142  8779          VBUSDATA        <^X24>,0
                        0143  8780          VBUSDATA        <^X25>,0
                    02  0144  8781          .BYTE   ^X2                             ;EMODF
                        0145  8782          VBUSDATA        <^X24>,0
                        0146  8783          VBUSDATA        <^X25>,0
                    02  0147  8784          .BYTE   ^X2
                        0148  8785          VBUSDATA        <^X24>,0
                        0149  8786          VBUSDATA        <^X25>,1
                    02  014A  8787          .BYTE   ^X2
                        014B  8788          VBUSDATA        <^X24>,0
                        014C  8789          VBUSDATA        <^X25>,1
                    02  014D  8790          .BYTE   ^X2
                        014E  8791          VBUSDATA        <^X24>,0
                        014F  8792          VBUSDATA        <^X25>,1
                    02  0150  8793          .BYTE   ^X2
                        0151  8794          VBUSDATA        <^X24>,0
                        0152  8795          VBUSDATA        <^X25>,0
                    02  0153  8796          .BYTE   ^X2                             ;POLYF
```

```
                    0154  8797          VBUSDATA        <^X24>,1
                    0155  8798          VBUSDATA        <^X25>,0
                02  0156  8799          .BYTE   ^X2
                    0157  8800          VBUSDATA        <^X24>,0
                    0158  8801          VBUSDATA        <^X25>,0
                02  0159  8802          .BYTE   ^X2
                    015A  8803          VBUSDATA        <^X24>,0
                    015B  8804          VBUSDATA        <^X25>,0
                02  015C  8805          .BYTE   ^X2
                    015D  8806          VBUSDATA        <^X24>,0
                    015E  8807          VBUSDATA        <^X25>,0
                02  015F  8808          .BYTE   ^X2
                    0160  8809          VBUSDATA        <^X24>,0
                    0161  8810          VBUSDATA        <^X25>,0
                02  0162  8811          .BYTE   ^X2                             ;CVTFD
                    0163  8812          VBUSDATA        <^X24>,1
                    0164  8813          VBUSDATA        <^X25>,1
                02  0165  8814          .BYTE   ^X2
                    0166  8815          VBUSDATA        <^X24>,0
                    0167  8816          VBUSDATA        <^X25>,0
                02  0168  8817          .BYTE   ^X2
                    0169  8818          VBUSDATA        <^X24>,0
                    016A  8819          VBUSDATA        <^X25>,0
                02  016B  8820          .BYTE   ^X2
                    016C  8821          VBUSDATA        <^X24>,0
                    016D  8822          VBUSDATA        <^X25>,0
                02  016E  8823          .BYTE   ^X2
                    016F  8824          VBUSDATA        <^X24>,0
                    0170  8825          VBUSDATA        <^X25>,0
                02  0171  8826          .BYTE   ^X2                             ;RESVRD
                    0172  8827          VBUSDATA        <^X24>,0
                    0173  8828          VBUSDATA        <^X25>,0
                02  0174  8829          .BYTE   ^X2
                    0175  8830          VBUSDATA        <^X24>,0
                    0176  8831          VBUSDATA        <^X25>,0
                02  0177  8832          .BYTE   ^X2
                    0178  8833          VBUSDATA        <^X24>,0
                    0179  8834          VBUSDATA        <^X25>,0
                02  017A  8835          .BYTE   ^X2
                    017B  8836          VBUSDATA        <^X24>,0
                    017C  8837          VBUSDATA        <^X25>,0
                02  017D  8838          .BYTE   ^X2
                    017E  8839          VBUSDATA        <^X24>,0
                    017F  8840          VBUSDATA        <^X25>,0
                02  0180  8841          .BYTE   ^X2                             ;ADAWI
                    0181  8842          VBUSDATA        <^X24>,1
                    0182  8843          VBUSDATA        <^X25>,0
                02  0183  8844          .BYTE   ^X2
                    0184  8845          VBUSDATA        <^X24>,0
                    0185  8846          VBUSDATA        <^X25>,0
                02  0186  8847          .BYTE   ^X2
                    0187  8848          VBUSDATA        <^X24>,0
                    0188  8849          VBUSDATA        <^X25>,0
                02  0189  8850          .BYTE   ^X2
                    018A  8851          VBUSDATA        <^X24>,0
```

```
                    018B  8852          VBUSDATA        <^X25>,0
                02  018C  8853          .BYTE   ^X2
                    018D  8854          VBUSDATA        <^X24>,0
                    018E  8855          VBUSDATA        <^X25>,0
                02  018F  8856          .BYTE   ^X2                             ;RESVRD
                    0190  8857          VBUSDATA        <^X24>,0
                    0191  8858          VBUSDATA        <^X25>,0
                02  0192  8859          .BYTE   ^X2
                    0193  8860          VBUSDATA        <^X24>,0
                    0194  8861          VBUSDATA        <^X25>,0
                02  0195  8862          .BYTE   ^X2
                    0196  8863          VBUSDATA        <^X24>,0
                    0197  8864          VBUSDATA        <^X25>,0
                02  0198  8865          .BYTE   ^X2
                    0199  8866          VBUSDATA        <^X24>,0
                    019A  8867          VBUSDATA        <^X25>,0
                02  019B  8868          .BYTE   ^X2
                    019C  8869          VBUSDATA        <^X24>,0
                    019D  8870          VBUSDATA        <^X25>,0
                02  019E  8871          .BYTE   ^X2                             ;RESVRD
                    019F  8872          VBUSDATA        <^X24>,0
                    01A0  8873          VBUSDATA        <^X25>,0
                02  01A1  8874          .BYTE   ^X2
                    01A2  8875          VBUSDATA        <^X24>,0
                    01A3  8876          VBUSDATA        <^X25>,0
                02  01A4  8877          .BYTE   ^X2
                    01A5  8878          VBUSDATA        <^X24>,0
                    01A6  8879          VBUSDATA        <^X25>,0
                02  01A7  8880          .BYTE   ^X2
                    01A8  8881          VBUSDATA        <^X24>,0
                    01A9  8882          VBUSDATA        <^X25>,0
                02  01AA  8883          .BYTE   ^X2
                    01AB  8884          VBUSDATA        <^X24>,0
                    01AC  8885          VBUSDATA        <^X25>,0
                02  01AD  8886          .BYTE   ^X2                             ;RESVRD
                    01AE  8887          VBUSDATA        <^X24>,0
                    01AF  8888          VBUSDATA        <^X25>,0
                02  01B0  8889          .BYTE   ^X2
                    01B1  8890          VBUSDATA        <^X24>,0
                    01B2  8891          VBUSDATA        <^X25>,0
                02  01B3  8892          .BYTE   ^X2
                    01B4  8893          VBUSDATA        <^X24>,0
                    01B5  8894          VBUSDATA        <^X25>,0
                02  01B6  8895          .BYTE   ^X2
                    01B7  8896          VBUSDATA        <^X24>,0
                    01B8  8897          VBUSDATA        <^X25>,0
                02  01B9  8898          .BYTE   ^X2
                    01BA  8899          VBUSDATA        <^X24>,0
                    01BB  8900          VBUSDATA        <^X25>,0
                02  01BC  8901          .BYTE   ^X2                             ;INSQHI
                    01BD  8902          VBUSDATA        <^X24>,1
                    01BE  8903          VBUSDATA        <^X25>,1
                02  01BF  8904          .BYTE   ^X2
                    01C0  8905          VBUSDATA        <^X24>,0
                    01C1  8906          VBUSDATA        <^X25>,0
```

```
                02  01C2  8907          .BYTE   ^X2
                    01C3  8908          VBUSDATA        <^X24>,0
                    01C4  8909          VBUSDATA        <^X25>,0
                02  01C5  8910          .BYTE   ^X2
                    01C6  8911          VBUSDATA        <^X24>,0
                    01C7  8912          VBUSDATA        <^X25>,0
                02  01C8  8913          .BYTE   ^X2
                    01C9  8914          VBUSDATA        <^X24>,0
                    01CA  8915          VBUSDATA        <^X25>,0
                02  01CB  8916          .BYTE   ^X2                             ;INSQTI
                    01CC  8917          VBUSDATA        <^X24>,1
                    01CD  8918          VBUSDATA        <^X25>,1
                02  01CE  8919          .BYTE   ^X2
                    01CF  8920          VBUSDATA        <^X24>,0
                    01D0  8921          VBUSDATA        <^X25>,0
                02  01D1  8922          .BYTE   ^X2
                    01D2  8923          VBUSDATA        <^X24>,0
                    01D3  8924          VBUSDATA        <^X25>,0
                02  01D4  8925          .BYTE   ^X2
                    01D5  8926          VBUSDATA        <^X24>,0
                    01D6  8927          VBUSDATA        <^X25>,0
                02  01D7  8928          .BYTE   ^X2
                    01D8  8929          VBUSDATA        <^X24>,0
                    01D9  8930          VBUSDATA        <^X25>,0
                02  01DA  8931          .BYTE   ^X2                             ;REMQHI
                    01DB  8932          VBUSDATA        <^X24>,0
                    01DC  8933          VBUSDATA        <^X25>,1
                02  01DD  8934          .BYTE   ^X2
                    01DE  8935          VBUSDATA        <^X24>,0
                    01DF  8936          VBUSDATA        <^X25>,0
                02  01E0  8937          .BYTE   ^X2
                    01E1  8938          VBUSDATA        <^X24>,0
                    01E2  8939          VBUSDATA        <^X25>,0
                02  01E3  8940          .BYTE   ^X2
                    01E4  8941          VBUSDATA        <^X24>,0
                    01E5  8942          VBUSDATA        <^X25>,0
                02  01E6  8943          .BYTE   ^X2
                    01E7  8944          VBUSDATA        <^X24>,0
                    01E8  8945          VBUSDATA        <^X25>,0
                02  01E9  8946          .BYTE   ^X2                             ;REMQTI
                    01EA  8947          VBUSDATA        <^X24>,0
                    01EB  8948          VBUSDATA        <^X25>,1
                02  01EC  8949          .BYTE   ^X2
                    01ED  8950          VBUSDATA        <^X24>,0
                    01EE  8951          VBUSDATA        <^X25>,0
                02  01EF  8952          .BYTE   ^X2
                    01F0  8953          VBUSDATA        <^X24>,0
                    01F1  8954          VBUSDATA        <^X25>,0
                02  01F2  8955          .BYTE   ^X2
                    01F3  8956          VBUSDATA        <^X24>,0
                    01F4  8957          VBUSDATA        <^X25>,0
                02  01F5  8958          .BYTE   ^X2
                    01F6  8959          VBUSDATA        <^X24>,0
                    01F7  8960          VBUSDATA        <^X25>,0
                02  01F8  8961          .BYTE   ^X2                             ;ADDD2
```

```
                    01F9  8962          VBUSDATA        <^X24>,1
                    01FA  8963          VBUSDATA        <^X25>,1
                02  01FB  8964          .BYTE   ^X2
                    01FC  8965          VBUSDATA        <^X24>,0
                    01FD  8966          VBUSDATA        <^X25>,0
                02  01FE  8967          .BYTE   ^X2
                    01FF  8968          VBUSDATA        <^X24>,0
                    0200  8969          VBUSDATA        <^X25>,0
                02  0201  8970          .BYTE   ^X2
                    0202  8971          VBUSDATA        <^X24>,0
                    0203  8972          VBUSDATA        <^X25>,0
                02  0204  8973          .BYTE   ^X2
                    0205  8974          VBUSDATA        <^X24>,0
                    0206  8975          VBUSDATA        <^X25>,0
                02  0207  8976          .BYTE   ^X2                             ;ADDD3
                    0208  8977          VBUSDATA        <^X24>,1
                    0209  8978          VBUSDATA        <^X25>,1
                02  020A  8979          .BYTE   ^X2
                    020B  8980          VBUSDATA        <^X24>,1
                    020C  8981          VBUSDATA        <^X25>,1
                02  020D  8982          .BYTE   ^X2
                    020E  8983          VBUSDATA        <^X24>,0
                    020F  8984          VBUSDATA        <^X25>,0
                02  0210  8985          .BYTE   ^X2
                    0211  8986          VBUSDATA        <^X24>,0
                    0212  8987          VBUSDATA        <^X25>,0
                02  0213  8988          .BYTE   ^X2
                    0214  8989          VBUSDATA        <^X24>,0
                    0215  8990          VBUSDATA        <^X25>,0
                02  0216  8991          .BYTE   ^X2                             ;SUBD2
                    0217  8992          VBUSDATA        <^X24>,1
                    0218  8993          VBUSDATA        <^X25>,1
                02  0219  8994          .BYTE   ^X2
                    021A  8995          VBUSDATA        <^X24>,0
                    021B  8996          VBUSDATA        <^X25>,0
                02  021C  8997          .BYTE   ^X2
                    021D  8998          VBUSDATA        <^X24>,0
                    021E  8999          VBUSDATA        <^X25>,0
                02  021F  9000          .BYTE   ^X2
                    0220  9001          VBUSDATA        <^X24>,0
                    0221  9002          VBUSDATA        <^X25>,0
                02  0222  9003          .BYTE   ^X2
                    0223  9004          VBUSDATA        <^X24>,0
                    0224  9005          VBUSDATA        <^X25>,0
                02  0225  9006          .BYTE   ^X2                             ;SUBD3
                    0226  9007          VBUSDATA        <^X24>,1
                    0227  9008          VBUSDATA        <^X25>,1
                02  0228  9009          .BYTE   ^X2
                    0229  9010          VBUSDATA        <^X24>,1
                    022A  9011          VBUSDATA        <^X25>,1
                02  022B  9012          .BYTE   ^X2
                    022C  9013          VBUSDATA        <^X24>,0
                    022D  9014          VBUSDATA        <^X25>,0
                02  022E  9015          .BYTE   ^X2
                    022F  9016          VBUSDATA        <^X24>,0
```

```
                                  0230  9017          VBUSDATA        <^X25>,0
                              02  0231  9018          .BYTE   ^X2
                                  0232  9019          VBUSDATA        <^X24>,0
                                  0233  9020          VBUSDATA        <^X25>,0
                              02  0234  9021          .BYTE   ^X2                                ;MULD2
                                  0235  9022          VBUSDATA        <^X24>,1
                                  0236  9023          VBUSDATA        <^X25>,1
                              02  0237  9024          .BYTE   ^X2
                                  0238  9025          VBUSDATA        <^X24>,0
                                  0239  9026          VBUSDATA        <^X25>,0
                              02  023A  9027          .BYTE   ^X2
                                  023B  9028          VBUSDATA        <^X24>,0
                                  023C  9029          VBUSDATA        <^X25>,0
                              02  023D  9030          .BYTE   ^X2
                                  023E  9031          VBUSDATA        <^X24>,0
                                  023F  9032          VBUSDATA        <^X25>,0
                              02  0240  9033          .BYTE   ^X2
                                  0241  9034          VBUSDATA        <^X24>,0
                                  0242  9035          VBUSDATA        <^X25>,0
                                  0243  9036
                                  0243  9037
                                  0243  9038
                                  0243  9039 BEGIN_CPU_DATA
                                  0002  9040
                                  0002  9041 DCS_DATA
                                  0001  9042
U 00, 7700,C800,0364,0300,0470,1801  0001  9043 ;% 00   NOP
U 01, 7700,4800,5BE4,0300,2470,1802  000C  9047 ;% 01   WCTRL/LOADCRAR,WB_R[TEMP0]                ;DISABLE ^P INTERRUPT
U 02, 7700,0800,5BE4,0304,2C70,1803  0017  9051 ;% 02   WCTRL/CONWRITE,WB_R[TEMP1]                ;...
U 03, 7700,9800,D370,030F,FA70,1804  0022  9055 ;% 03   IPL_[1F]                                  ;SET IPL TO 1F
U 04, 5700,4840,0364,0308,0470,1805  002D  9059 ;% 04   R[TEMP2]_RDM                              ;MOVE LOOP COUNT TO RTEMP2
U 05, 7700,D800,EF64,0301,0470,1806  0038  9063 ;% 05   PL_[2]                                    ;SET PLATCH TO 2
U 06, 7700,C800,8B71,0308,0470,1807  0043  9067 ;% 06   Q_R[TEMP2].RL.P                           ;ROTATE LOOP COUNT LEFT 2
U 07, 7700,4800,03A4,03D8,4870,1808  004E  9071 ;% 07   WCTRL/PC_WB,WB_Q                          ;FORCE PREFETCH TO HAPPEN
U 08, 7700,4800,0364,1700,0470,1809  0059  9075 ;% 08   IRD1TEST                                  ;LOAD IR REGISTER
U 09, 7600,4800,0364,0300,0470,180A  0064  9079 ;% 09   DTYPE/IDEP,STROBE.V.BUS                   ;LOAD DSIZE INTO VBUS
U 0A, 7300,C800,0364,0300,0470,180B  006F  9083 ;% 0A   NOP,STOP.CLOCK                            ;STOP THE CPU CLOCK
U 0B, 7700,4800,0364,0300,0470,180C  007A  9087 ;% 0B   NOP
U 0C, 7700,C800,0364,1B00,0470,180D  0085  9091 ;% 0C   LOD INC BRA?                              ;INCREMENT THE IRD COUNTER
U 0D, 7600,C800,0364,0300,0470,180E  0090  9095 ;% 0D   DTYPE/IDEP,STROBE.V.BUS                   ;LOAD DSIZE INTO VBUS
U 0E, 7300,4800,0364,0300,0470,180F  009B  9099 ;% 0E   NOP,STOP.CLOCK                            ;STOP THE CPU CLOCK
                                  00A6  9103
                                  00A6  9104 RTEMP_DATA
                                  0001  9105
                          00C00000  0001  9106          .LONG   ^X00C00000                        ;DISABLE ^P INTERRUPT
                          00400000  0005  9107          .LONG   ^X00400000                        ;...
                                  0009  9108
                                  0009  9109 CACHE_DATA
                                  0001  9110
                          0000004C  0001  9111          .LONG   ^X0000004C                        ;CVTBF
                          0000004D  0005  9112          .LONG   ^X0000004D                        ;CVTWF
                          0000004E  0009  9113          .LONG   ^X0000004E                        ;CVTLF
                          0000004F  000D  9114          .LONG   ^X0000004F                        ;ACBF
                          00000050  0011  9115          .LONG   ^X00000050                        ;MOVF
                          00000051  0015  9116          .LONG   ^X00000051                        ;CMPF
```

```
00000052  0019  9117          .LONG   ^X00000052                      ;MNEGF
00000053  001D  9118          .LONG   ^X00000053                      ;TSTF
00000054  0021  9119          .LONG   ^X00000054                      ;EMODF
00000055  0025  9120          .LONG   ^X00000055                      ;POLYF
00000056  0029  9121          .LONG   ^X00000056                      ;CVTFD
00000057  002D  9122          .LONG   ^X00000057                      ;RESVRD
00000058  0031  9123          .LONG   ^X00000058                      ;ADAWI
00000059  0035  9124          .LONG   ^X00000059                      ;RESVRD
0000005A  0039  9125          .LONG   ^X0000005A                      ;RESVRD
0000005B  003D  9126          .LONG   ^X0000005B                      ;RESVRD
0000005C  0041  9127          .LONG   ^X0000005C                      ;RESVRD
0000005D  0045  9128          .LONG   ^X0000005D                      ;RESVRD
0000005E  0049  9129          .LONG   ^X0000005E                      ;REMQHI
0000005F  004D  9130          .LONG   ^X0000005F                      ;REMQTI
00000060  0051  9131          .LONG   ^X00000060                      ;ADDD2
00000061  0055  9132          .LONG   ^X00000061                      ;ADDD3
00000062  0059  9133          .LONG   ^X00000062                      ;SUBD2
00000063  005D  9134          .LONG   ^X00000063                      ;SUBD3
00000064  0061  9135          .LONG   ^X00000064                      ;MULD2
00000000  0065  9136          .LONG   ^X00000000
          0069  9137
          0069  9138 END_CPU_DATA
          0243  9139
          0243  9140
          0243  9141 END_TEST_DATA
          0243
          0243          ;-----------------------------------------------------------------------------
          0243  9142 ENDTEST
```

```
                        0021        .SBTTL  "TEST   088     NATIVE MODE DSIZE ROM TEST 6
                        0021  9144 ;*******************************************************************************
                        0021  9145 ;++
                        0021  9146 ;
                        0021  9147 ; FUNCTIONAL DESCRIPTION:
                        0021  9148 ;
                        0021  9149 ;       THIS IS ONE IN A SERIES OF 14 TESTS THAT CHECK THE CONTENTS OF THE
                        0021  9150 ;       DSIZE ROM.
                        0021  9151 ;
                        0021  9152 ;
                        0021  9153 ; TEST ALGORITHM:
                        0021  9154 ;
                        0021  9155 ;       1.  CREATE AN INDEX (K) TO INDEX RESULTS TABLE
                        0021  9156 ;       2.  CREATE LOOP (I) OF 25
                        0021  9157 ;       3.  SAVE LOOP COUNT IN 1$
                        0021  9158 ;       4.  MOVE LOOP COUNT TO WDREG
                        0021  9159 ;       5.  EXECUTE DCS PROGRAM
                        0021  9160 ;               a.  DISABLE ^P INTERRUPTS
                        0021  9161 ;               b.  SET IPL TO 1F
                        0021  9162 ;               c.  MOVE LOOP COUNT TO RTEMP2
                        0021  9163 ;               d.  ROTATE LOOP COUNT LEFT 2 TO FORM CACHE ADDRESS
                        0021  9164 ;               e.  LOAD THIS ADDRESS INTO PC TO CAUSE PREFETCH
                        0021  9165 ;               f.  PERFORM IRD1TEST TO CAUSE IR TO LOAD INSTRUCTION
                        0021  9166 ;               g.  SET DTYPE TO INSTURCTION DEPENDENT AND STROBE THE VBUS
                        0021  9167 ;       6.  TEST VBUS FOR CORRECT DSIZE
                        0021  9168 ;       7.  IF NO ERROR CREATE LOOP (J) OF 5
                        0021  9169 ;       8.  EXECUTE DCS PROGRAM
                        0021  9170 ;               a.  INCREMENT THE IRD COUNTER
                        0021  9171 ;               b.  SET DTYPE TO INSTRUCTION DEPENDENT AND STROBE THE VBUS
                        0021  9172 ;       9.  TEST VBUS FOR CORRECT DSIZE
                        0021  9173 ;      10.  INCREMENT INDEX K BY ONE
                        0021  9174 ;      11.  IF NO ERROR GO TO STEP 8 UNTIL LOOP J IS EXHAUSTED
                        0021  9175 ;      12.  GO TO STEP 3 UNTIL LOOP I IS EXHAUSTED
                        0021  9176 ;
                        0021  9177 ;
                        0021  9178 ; TEST PATTERNS:
                        0021  9179 ;
                        0021  9180 ;       LOOP COUNT I            INSTRUCTION
                        0021  9181 ;            1                     MULD3
                        0021  9182 ;            2                     DIVD2
                        0021  9183 ;            3                     DIVD3
                        0021  9184 ;            4                     CVTDB
                        0021  9185 ;            5                     CVTDW
                        0021  9186 ;            6                     CVTDL
                        0021  9187 ;            7                     CVTRDL
                        0021  9188 ;            8                     CVTBD
                        0021  9189 ;            9                     CVTWD
                        0021  9190 ;            A                     CVTLD
                        0021  9191 ;            B                     ACBD
                        0021  9192 ;            C                     MOVD
                        0021  9193 ;            D                     CMPD
                        0021  9194 ;            E                     MNEGD
                        0021  9195 ;            F                     TSTD
                        0021  9196 ;           10                     EMODD
                        0021  9197 ;           11                     POLYD
```

```
0021  ;          12                    CVTDF
0021  ;          13                    RESVRD
0021  ;          14                    ASHL
0021  ;          15                    ASHQ
0021  ;          16                    EMUL
0021  ;          17                    EDIV
0021  ;          18                    CLRQ
0021  ;          19                    MOVQ
0021  ;
0021  ;
0021  ; LOGIC DESCRIPTION:
0021  ;
0021  ;       THIS TEST ALONG WITH THE OTHER 13 TESTS IN THIS SERIES CHECK ALL THE
0021  ;       LOCATIONS BLASTED INTO THE DSIZE ROM.  ANY FALIURES SHOULD BE CONFINED
0021  ;       TO THE ROM.
0021  ;
0021  ;
0021  ; ASSUMPTIONS:
0021  ;
0021  ;       THIS TEST ASSUMES THE PREFETCH LOGIC AND INSTRUCTION REGISTER ARE
0021  ;       OPERATIONAL.
0021  ;
0021  ;
0021  ; ERROR DESCRIPTION:
0021  ;
0021  ;       EXPECTED VBUS DATA (DSIZE)
0021  ;       RECEIVED VBUS DATA
0021  ;       LOOP COUNT I
0021  ;       LOOP COUNT J
0021  ;       LOOP COUNT K
0021  ;               (BITS <7:0> = VBUS BIT POSITION)
0021  ;               (BITS <8> = BIT STATE)
0021  ;
0021  ;--
0021  ;*******************************************************************************
0021
0021
0021          INITIALIZE                                  ;INIT THE CPU
0023          LDINDEX         K,1                         ;INIT INDEX K TO 1
002A          LOOP            I,1,25                      ;CREATE A TEST LOOP OF 50
0031          SAVEINDEX       I,1$                        ;SAVE LOOP COUNT IN 1$
0036          LOADREG         WDREG,1$,,L                 ;LOAD LOOP COUNT INTO WDREG
003E          ERRLOOP                                     ;ERROR LOOP POINT
0040          FETCH           0                           ;START DCS PROGRAM
0045          BRSTCLK         <^X0A>,INH                  ;END DCS PROGRAM
0049          CMPVBUS         2$,I                        ;TEST DSIZE
004E          IFERROR                                     ;ROM FAILURE
0053          LOOP            J,1,5                       ;CREATE TEST LOOP OF 5
005A          FETCH           <^X0B>                      ;START DCS PROGRAM
005F          BRSTCLK         <^X0E>,INH                  ;END DCS PROGRAM
0063          CMPVBUS         3$,K                        ;TEST DSIZE
0068          IFERROR                                     ;ROM FAILURE
006D          INCINDEX        K                           ;INCREMENT INDEX K BY ONE
0070          ENDLOOP         J                           ;END TEST LOOP J
0073          ENDLOOP         I                           ;END TEST LOOP I
```

```
                    0076  9253          SKIP                                    ;GO TO NEXT TEST
                    007D  9254
                    007D  9255
                    007D  9256 BEGIN_TEST_DATA
                    007D
                    007D          ;-----------------------------------------------------------------------
                    007D  9257
           00000000 007D  9258 1$:      .LONG   ^X00000000                      ;LOOP COUNT STORAGE
                 02 0081  9259 2$:      .BYTE   ^X2                             ;RESULTS TABLE (IRD 0)
                    0082  9260          VBUSDATA        <^X24>,1                ;MULD3
                    0083  9261          VBUSDATA        <^X25>,1
                 02 0084  9262          .BYTE   ^X2                             ;DIVD2
                    0085  9263          VBUSDATA        <^X24>,1
                    0086  9264          VBUSDATA        <^X25>,1
                 02 0087  9265          .BYTE   ^X2                             ;DIVD3
                    0088  9266          VBUSDATA        <^X24>,1
                    0089  9267          VBUSDATA        <^X25>,1
                 02 008A  9268          .BYTE   ^X2                             ;CVTDB
                    008B  9269          VBUSDATA        <^X24>,1
                    008C  9270          VBUSDATA        <^X25>,1
                 02 008D  9271          .BYTE   ^X2                             ;CVTDW
                    008E  9272          VBUSDATA        <^X24>,1
                    008F  9273          VBUSDATA        <^X25>,1
                 02 0090  9274          .BYTE   ^X2                             ;CVTDL
                    0091  9275          VBUSDATA        <^X24>,1
                    0092  9276          VBUSDATA        <^X25>,1
                 02 0093  9277          .BYTE   ^X2                             ;CVTRDL
                    0094  9278          VBUSDATA        <^X24>,1
                    0095  9279          VBUSDATA        <^X25>,1
                 02 0096  9280          .BYTE   ^X2                             ;CVTBD
                    0097  9281          VBUSDATA        <^X24>,0
                    0098  9282          VBUSDATA        <^X25>,0
                 02 0099  9283          .BYTE   ^X2                             ;CVTWD
                    009A  9284          VBUSDATA        <^X24>,1
                    009B  9285          VBUSDATA        <^X25>,0
                 02 009C  9286          .BYTE   ^X2                             ;CVTLD
                    009D  9287          VBUSDATA        <^X24>,0
                    009E  9288          VBUSDATA        <^X25>,1
                 02 009F  9289          .BYTE   ^X2                             ;ACBD
                    00A0  9290          VBUSDATA        <^X24>,1
                    00A1  9291          VBUSDATA        <^X25>,1
                 02 00A2  9292          .BYTE   ^X2                             ;MOVD
                    00A3  9293          VBUSDATA        <^X24>,1
                    00A4  9294          VBUSDATA        <^X25>,1
                 02 00A5  9295          .BYTE   ^X2                             ;CMPD
                    00A6  9296          VBUSDATA        <^X24>,1
                    00A7  9297          VBUSDATA        <^X25>,1
                 02 00A8  9298          .BYTE   ^X2                             ;MNEGD
                    00A9  9299          VBUSDATA        <^X24>,1
                    00AA  9300          VBUSDATA        <^X25>,1
                 02 00AB  9301          .BYTE   ^X2                             ;TSTD
                    00AC  9302          VBUSDATA        <^X24>,1
                    00AD  9303          VBUSDATA        <^X25>,1
                 02 00AE  9304          .BYTE   ^X2                             ;EMODD
                    00AF  9305          VBUSDATA        <^X24>,1
```

```
                    00B0  9306          VBUSDATA        <^X25>,1
02                  00B1  9307          .BYTE   ^X2                             ;POLYD
                    00B2  9308          VBUSDATA        <^X24>,1
                    00B3  9309          VBUSDATA        <^X25>,1
02                  00B4  9310          .BYTE   ^X2                             ;CVTDF
                    00B5  9311          VBUSDATA        <^X24>,1
                    00B6  9312          VBUSDATA        <^X25>,1
02                  00B7  9313          .BYTE   ^X2                             ;RESVRD
                    00B8  9314          VBUSDATA        <^X24>,0
                    00B9  9315          VBUSDATA        <^X25>,0
02                  00BA  9316          .BYTE   ^X2                             ;ASHL
                    00BB  9317          VBUSDATA        <^X24>,0
                    00BC  9318          VBUSDATA        <^X25>,0
02                  00BD  9319          .BYTE   ^X2                             ;ASHQ
                    00BE  9320          VBUSDATA        <^X24>,0
                    00BF  9321          VBUSDATA        <^X25>,0
02                  00C0  9322          .BYTE   ^X2                             ;EMUL
                    00C1  9323          VBUSDATA        <^X24>,0
                    00C2  9324          VBUSDATA        <^X25>,1
02                  00C3  9325          .BYTE   ^X2                             ;EDIV
                    00C4  9326          VBUSDATA        <^X24>,0
                    00C5  9327          VBUSDATA        <^X25>,1
02                  00C6  9328          .BYTE   ^X2                             ;CLRQ
                    00C7  9329          VBUSDATA        <^X24>,1
                    00C8  9330          VBUSDATA        <^X25>,1
02                  00C9  9331          .BYTE   ^X2                             ;MOVQ
                    00CA  9332          VBUSDATA        <^X24>,1
                    00CB  9333          VBUSDATA        <^X25>,1
                    00CC  9334
02                  00CC  9335 3$:      .BYTE   ^X2                             ;RESULTS TABLE (IRD 1 TO 5)
                    00CD  9336          VBUSDATA        <^X24>,1                ;MULD3
                    00CE  9337          VBUSDATA        <^X25>,1
02                  00CF  9338          .BYTE   ^X2
                    00D0  9339          VBUSDATA        <^X24>,1
                    00D1  9340          VBUSDATA        <^X25>,1
02                  00D2  9341          .BYTE   ^X2
                    00D3  9342          VBUSDATA        <^X24>,0
                    00D4  9343          VBUSDATA        <^X25>,0
02                  00D5  9344          .BYTE   ^X2
                    00D6  9345          VBUSDATA        <^X24>,0
                    00D7  9346          VBUSDATA        <^X25>,0
02                  00D8  9347          .BYTE   ^X2
                    00D9  9348          VBUSDATA        <^X24>,0
                    00DA  9349          VBUSDATA        <^X25>,0
02                  00DB  9350          .BYTE   ^X2                             ;DIVD2
                    00DC  9351          VBUSDATA        <^X24>,1
                    00DD  9352          VBUSDATA        <^X25>,1
02                  00DE  9353          .BYTE   ^X2
                    00DF  9354          VBUSDATA        <^X24>,0
                    00E0  9355          VBUSDATA        <^X25>,0
02                  00E1  9356          .BYTE   ^X2
                    00E2  9357          VBUSDATA        <^X24>,0
                    00E3  9358          VBUSDATA        <^X25>,0
02                  00E4  9359          .BYTE   ^X2
                    00E5  9360          VBUSDATA        <^X24>,0
```

```
                    00E6  9361          VBUSDATA        <^X25>,0
                02  00E7  9362          .BYTE   ^X2
                    00E8  9363          VBUSDATA        <^X24>,0
                    00E9  9364          VBUSDATA        <^X25>,0
                02  00EA  9365          .BYTE   ^X2                             ;DIVD3
                    00EB  9366          VBUSDATA        <^X24>,1
                    00EC  9367          VBUSDATA        <^X25>,1
                02  00ED  9368          .BYTE   ^X2
                    00EE  9369          VBUSDATA        <^X24>,1
                    00EF  9370          VBUSDATA        <^X25>,1
                02  00F0  9371          .BYTE   ^X2
                    00F1  9372          VBUSDATA        <^X24>,0
                    00F2  9373          VBUSDATA        <^X25>,0
                02  00F3  9374          .BYTE   ^X2
                    00F4  9375          VBUSDATA        <^X24>,0
                    00F5  9376          VBUSDATA        <^X25>,0
                02  00F6  9377          .BYTE   ^X2
                    00F7  9378          VBUSDATA        <^X24>,0
                    00F8  9379          VBUSDATA        <^X25>,0
                02  00F9  9380          .BYTE   ^X2                             ;CVTDB
                    00FA  9381          VBUSDATA        <^X24>,0
                    00FB  9382          VBUSDATA        <^X25>,0
                02  00FC  9383          .BYTE   ^X2
                    00FD  9384          VBUSDATA        <^X24>,0
                    00FE  9385          VBUSDATA        <^X25>,0
                02  00FF  9386          .BYTE   ^X2
                    0100  9387          VBUSDATA        <^X24>,0
                    0101  9388          VBUSDATA        <^X25>,0
                02  0102  9389          .BYTE   ^X2
                    0103  9390          VBUSDATA        <^X24>,0
                    0104  9391          VBUSDATA        <^X25>,0
                02  0105  9392          .BYTE   ^X2
                    0106  9393          VBUSDATA        <^X24>,0
                    0107  9394          VBUSDATA        <^X25>,0
                02  0108  9395          .BYTE   ^X2                             ;CVTDW
                    0109  9396          VBUSDATA        <^X24>,1
                    010A  9397          VBUSDATA        <^X25>,0
                02  010B  9398          .BYTE   ^X2
                    010C  9399          VBUSDATA        <^X24>,0
                    010D  9400          VBUSDATA        <^X25>,0
                02  010E  9401          .BYTE   ^X2
                    010F  9402          VBUSDATA        <^X24>,0
                    0110  9403          VBUSDATA        <^X25>,0
                02  0111  9404          .BYTE   ^X2
                    0112  9405          VBUSDATA        <^X24>,0
                    0113  9406          VBUSDATA        <^X25>,0
                02  0114  9407          .BYTE   ^X2
                    0115  9408          VBUSDATA        <^X24>,0
                    0116  9409          VBUSDATA        <^X25>,0
                02  0117  9410          .BYTE   ^X2                             ;CVTDL
                    0118  9411          VBUSDATA        <^X24>,0
                    0119  9412          VBUSDATA        <^X25>,1
                02  011A  9413          .BYTE   ^X2
                    011B  9414          VBUSDATA        <^X24>,0
                    011C  9415          VBUSDATA        <^X25>,0
```

```
                        02  011D  9416          .BYTE   ^X2
                            011E  9417          VBUSDATA        <^X24>,0
                            011F  9418          VBUSDATA        <^X25>,0
                        02  0120  9419          .BYTE   ^X2
                            0121  9420          VBUSDATA        <^X24>,0
                            0122  9421          VBUSDATA        <^X25>,0
                        02  0123  9422          .BYTE   ^X2
                            0124  9423          VBUSDATA        <^X24>,0
                            0125  9424          VBUSDATA        <^X25>,0
                        02  0126  9425          .BYTE   ^X2                             ;CVTRDL
                            0127  9426          VBUSDATA        <^X24>,0
                            0128  9427          VBUSDATA        <^X25>,1
                        02  0129  9428          .BYTE   ^X2
                            012A  9429          VBUSDATA        <^X24>,0
                            012B  9430          VBUSDATA        <^X25>,0
                        02  012C  9431          .BYTE   ^X2
                            012D  9432          VBUSDATA        <^X24>,0
                            012E  9433          VBUSDATA        <^X25>,0
                        02  012F  9434          .BYTE   ^X2
                            0130  9435          VBUSDATA        <^X24>,0
                            0131  9436          VBUSDATA        <^X25>,0
                        02  0132  9437          .BYTE   ^X2
                            0133  9438          VBUSDATA        <^X24>,0
                            0134  9439          VBUSDATA        <^X25>,0
                        02  0135  9440          .BYTE   ^X2                             ;CVTBD
                            0136  9441          VBUSDATA        <^X24>,1
                            0137  9442          VBUSDATA        <^X25>,1
                        02  0138  9443          .BYTE   ^X2
                            0139  9444          VBUSDATA        <^X24>,0
                            013A  9445          VBUSDATA        <^X25>,0
                        02  013B  9446          .BYTE   ^X2
                            013C  9447          VBUSDATA        <^X24>,0
                            013D  9448          VBUSDATA        <^X25>,0
                        02  013E  9449          .BYTE   ^X2
                            013F  9450          VBUSDATA        <^X24>,0
                            0140  9451          VBUSDATA        <^X25>,0
                        02  0141  9452          .BYTE   ^X2
                            0142  9453          VBUSDATA        <^X24>,0
                            0143  9454          VBUSDATA        <^X25>,0
                        02  0144  9455          .BYTE   ^X2                             ;CVTWD
                            0145  9456          VBUSDATA        <^X24>,1
                            0146  9457          VBUSDATA        <^X25>,1
                        02  0147  9458          .BYTE   ^X2
                            0148  9459          VBUSDATA        <^X24>,0
                            0149  9460          VBUSDATA        <^X25>,0
                        02  014A  9461          .BYTE   ^X2
                            014B  9462          VBUSDATA        <^X24>,0
                            014C  9463          VBUSDATA        <^X25>,0
                        02  014D  9464          .BYTE   ^X2
                            014E  9465          VBUSDATA        <^X24>,0
                            014F  9466          VBUSDATA        <^X25>,0
                        02  0150  9467          .BYTE   ^X2
                            0151  9468          VBUSDATA        <^X24>,0
                            0152  9469          VBUSDATA        <^X25>,0
                        02  0153  9470          .BYTE   ^X2                             ;CVTLD
```

```
                    0154  9471          VBUSDATA        <^X24>,1
                    0155  9472          VBUSDATA        <^X25>,1
                02  0156  9473          .BYTE   ^X2
                    0157  9474          VBUSDATA        <^X24>,0
                    0158  9475          VBUSDATA        <^X25>,0
                02  0159  9476          .BYTE   ^X2
                    015A  9477          VBUSDATA        <^X24>,0
                    015B  9478          VBUSDATA        <^X25>,0
                02  015C  9479          .BYTE   ^X2
                    015D  9480          VBUSDATA        <^X24>,0
                    015E  9481          VBUSDATA        <^X25>,0
                02  015F  9482          .BYTE   ^X2
                    0160  9483          VBUSDATA        <^X24>,0
                    0161  9484          VBUSDATA        <^X25>,0
                02  0162  9485          .BYTE   ^X2                             ;ACBD
                    0163  9486          VBUSDATA        <^X24>,1
                    0164  9487          VBUSDATA        <^X25>,1
                02  0165  9488          .BYTE   ^X2
                    0166  9489          VBUSDATA        <^X24>,1
                    0167  9490          VBUSDATA        <^X25>,1
                02  0168  9491          .BYTE   ^X2
                    0169  9492          VBUSDATA        <^X24>,0
                    016A  9493          VBUSDATA        <^X25>,0
                02  016B  9494          .BYTE   ^X2
                    016C  9495          VBUSDATA        <^X24>,0
                    016D  9496          VBUSDATA        <^X25>,0
                02  016E  9497          .BYTE   ^X2
                    016F  9498          VBUSDATA        <^X24>,0
                    0170  9499          VBUSDATA        <^X25>,0
                02  0171  9500          .BYTE   ^X2                             ;MOVD
                    0172  9501          VBUSDATA        <^X24>,1
                    0173  9502          VBUSDATA        <^X25>,1
                02  0174  9503          .BYTE   ^X2
                    0175  9504          VBUSDATA        <^X24>,0
                    0176  9505          VBUSDATA        <^X25>,0
                02  0177  9506          .BYTE   ^X2
                    0178  9507          VBUSDATA        <^X24>,0
                    0179  9508          VBUSDATA        <^X25>,0
                02  017A  9509          .BYTE   ^X2
                    017B  9510          VBUSDATA        <^X24>,0
                    017C  9511          VBUSDATA        <^X25>,0
                02  017D  9512          .BYTE   ^X2
                    017E  9513          VBUSDATA        <^X24>,0
                    017F  9514          VBUSDATA        <^X25>,0
                02  0180  9515          .BYTE   ^X2                             ;CMPD
                    0181  9516          VBUSDATA        <^X24>,1
                    0182  9517          VBUSDATA        <^X25>,1
                02  0183  9518          .BYTE   ^X2
                    0184  9519          VBUSDATA        <^X24>,0
                    0185  9520          VBUSDATA        <^X25>,0
                02  0186  9521          .BYTE   ^X2
                    0187  9522          VBUSDATA        <^X24>,0
                    0188  9523          VBUSDATA        <^X25>,0
                02  0189  9524          .BYTE   ^X2
                    018A  9525          VBUSDATA        <^X24>,0
```

```
                018B  9526          VBUSDATA        <^X25>,0
02              018C  9527          .BYTE   ^X2
                018D  9528          VBUSDATA        <^X24>,0
                018E  9529          VBUSDATA        <^X25>,0
02              018F  9530          .BYTE   ^X2                             ;MNEGD
                0190  9531          VBUSDATA        <^X24>,1
                0191  9532          VBUSDATA        <^X25>,1
02              0192  9533          .BYTE   ^X2
                0193  9534          VBUSDATA        <^X24>,0
                0194  9535          VBUSDATA        <^X25>,0
02              0195  9536          .BYTE   ^X2
                0196  9537          VBUSDATA        <^X24>,0
                0197  9538          VBUSDATA        <^X25>,0
02              0198  9539          .BYTE   ^X2
                0199  9540          VBUSDATA        <^X24>,0
                019A  9541          VBUSDATA        <^X25>,0
02              019B  9542          .BYTE   ^X2
                019C  9543          VBUSDATA        <^X24>,0
                019D  9544          VBUSDATA        <^X25>,0
02              019E  9545          .BYTE   ^X2                             ;TSTD
                019F  9546          VBUSDATA        <^X24>,0
                01A0  9547          VBUSDATA        <^X25>,0
02              01A1  9548          .BYTE   ^X2
                01A2  9549          VBUSDATA        <^X24>,0
                01A3  9550          VBUSDATA        <^X25>,0
02              01A4  9551          .BYTE   ^X2
                01A5  9552          VBUSDATA        <^X24>,0
                01A6  9553          VBUSDATA        <^X25>,0
02              01A7  9554          .BYTE   ^X2
                01A8  9555          VBUSDATA        <^X24>,0
                01A9  9556          VBUSDATA        <^X25>,0
02              01AA  9557          .BYTE   ^X2
                01AB  9558          VBUSDATA        <^X24>,0
                01AC  9559          VBUSDATA        <^X25>,0
02              01AD  9560          .BYTE   ^X2                             ;EMODD
                01AE  9561          VBUSDATA        <^X24>,0
                01AF  9562          VBUSDATA        <^X25>,0
02              01B0  9563          .BYTE   ^X2
                01B1  9564          VBUSDATA        <^X24>,1
                01B2  9565          VBUSDATA        <^X25>,1
02              01B3  9566          .BYTE   ^X2
                01B4  9567          VBUSDATA        <^X24>,0
                01B5  9568          VBUSDATA        <^X25>,1
02              01B6  9569          .BYTE   ^X2
                01B7  9570          VBUSDATA        <^X24>,1
                01B8  9571          VBUSDATA        <^X25>,1
02              01B9  9572          .BYTE   ^X2
                01BA  9573          VBUSDATA        <^X24>,0
                01BB  9574          VBUSDATA        <^X25>,0
02              01BC  9575          .BYTE   ^X2                             ;POLYD
                01BD  9576          VBUSDATA        <^X24>,1
                01BE  9577          VBUSDATA        <^X25>,0
02              01BF  9578          .BYTE   ^X2
                01C0  9579          VBUSDATA        <^X24>,0
                01C1  9580          VBUSDATA        <^X25>,0
```

```
                    02  01C2  9581          .BYTE   ^X2
                        01C3  9582          VBUSDATA        <^X24>,0
                        01C4  9583          VBUSDATA        <^X25>,0
                    02  01C5  9584          .BYTE   ^X2
                        01C6  9585          VBUSDATA        <^X24>,0
                        01C7  9586          VBUSDATA        <^X25>,0
                    02  01C8  9587          .BYTE   ^X2
                        01C9  9588          VBUSDATA        <^X24>,0
                        01CA  9589          VBUSDATA        <^X25>,0
                    02  01CB  9590          .BYTE   ^X2                             ;CVTDF
                        01CC  9591          VBUSDATA        <^X24>,0
                        01CD  9592          VBUSDATA        <^X25>,1
                    02  01CE  9593          .BYTE   ^X2
                        01CF  9594          VBUSDATA        <^X24>,0
                        01D0  9595          VBUSDATA        <^X25>,0
                    02  01D1  9596          .BYTE   ^X2
                        01D2  9597          VBUSDATA        <^X24>,0
                        01D3  9598          VBUSDATA        <^X25>,0
                    02  01D4  9599          .BYTE   ^X2
                        01D5  9600          VBUSDATA        <^X24>,0
                        01D6  9601          VBUSDATA        <^X25>,0
                    02  01D7  9602          .BYTE   ^X2
                        01D8  9603          VBUSDATA        <^X24>,0
                        01D9  9604          VBUSDATA        <^X25>,0
                    02  01DA  9605          .BYTE   ^X2                             ;RESVRD
                        01DB  9606          VBUSDATA        <^X24>,0
                        01DC  9607          VBUSDATA        <^X25>,0
                    02  01DD  9608          .BYTE   ^X2
                        01DE  9609          VBUSDATA        <^X24>,0
                        01DF  9610          VBUSDATA        <^X25>,0
                    02  01E0  9611          .BYTE   ^X2
                        01E1  9612          VBUSDATA        <^X24>,0
                        01E2  9613          VBUSDATA        <^X25>,0
                    02  01E3  9614          .BYTE   ^X2
                        01E4  9615          VBUSDATA        <^X24>,0
                        01E5  9616          VBUSDATA        <^X25>,0
                    02  01E6  9617          .BYTE   ^X2
                        01E7  9618          VBUSDATA        <^X24>,0
                        01E8  9619          VBUSDATA        <^X25>,0
                    02  01E9  9620          .BYTE   ^X2                             ;ASHL
                        01EA  9621          VBUSDATA        <^X24>,0
                        01EB  9622          VBUSDATA        <^X25>,1
                    02  01EC  9623          .BYTE   ^X2
                        01ED  9624          VBUSDATA        <^X24>,0
                        01EE  9625          VBUSDATA        <^X25>,1
                    02  01EF  9626          .BYTE   ^X2
                        01F0  9627          VBUSDATA        <^X24>,0
                        01F1  9628          VBUSDATA        <^X25>,0
                    02  01F2  9629          .BYTE   ^X2
                        01F3  9630          VBUSDATA        <^X24>,0
                        01F4  9631          VBUSDATA        <^X25>,0
                    02  01F5  9632          .BYTE   ^X2
                        01F6  9633          VBUSDATA        <^X24>,0
                        01F7  9634          VBUSDATA        <^X25>,0
                    02  01F8  9635          .BYTE   ^X2                             ;ASHQ
```

```
                    01F9  9636          VBUSDATA        <^X24>,1
                    01FA  9637          VBUSDATA        <^X25>,1
                02  01FB  9638          .BYTE   ^X2
                    01FC  9639          VBUSDATA        <^X24>,1
                    01FD  9640          VBUSDATA        <^X25>,1
                02  01FE  9641          .BYTE   ^X2
                    01FF  9642          VBUSDATA        <^X24>,0
                    0200  9643          VBUSDATA        <^X25>,0
                02  0201  9644          .BYTE   ^X2
                    0202  9645          VBUSDATA        <^X24>,0
                    0203  9646          VBUSDATA        <^X25>,0
                02  0204  9647          .BYTE   ^X2
                    0205  9648          VBUSDATA        <^X24>,0
                    0206  9649          VBUSDATA        <^X25>,0
                02  0207  9650          .BYTE   ^X2                             ;EMUL
                    0208  9651          VBUSDATA        <^X24>,0
                    0209  9652          VBUSDATA        <^X25>,1
                02  020A  9653          .BYTE   ^X2
                    020B  9654          VBUSDATA        <^X24>,0
                    020C  9655          VBUSDATA        <^X25>,1
                02  020D  9656          .BYTE   ^X2
                    020E  9657          VBUSDATA        <^X24>,1
                    020F  9658          VBUSDATA        <^X25>,1
                02  0210  9659          .BYTE   ^X2
                    0211  9660          VBUSDATA        <^X24>,0
                    0212  9661          VBUSDATA        <^X25>,0
                02  0213  9662          .BYTE   ^X2
                    0214  9663          VBUSDATA        <^X24>,0
                    0215  9664          VBUSDATA        <^X25>,0
                02  0216  9665          .BYTE   ^X2                             ;EDIV
                    0217  9666          VBUSDATA        <^X24>,1
                    0218  9667          VBUSDATA        <^X25>,1
                02  0219  9668          .BYTE   ^X2
                    021A  9669          VBUSDATA        <^X24>,0
                    021B  9670          VBUSDATA        <^X25>,1
                02  021C  9671          .BYTE   ^X2
                    021D  9672          VBUSDATA        <^X24>,0
                    021E  9673          VBUSDATA        <^X25>,1
                02  021F  9674          .BYTE   ^X2
                    0220  9675          VBUSDATA        <^X24>,0
                    0221  9676          VBUSDATA        <^X25>,0
                02  0222  9677          .BYTE   ^X2
                    0223  9678          VBUSDATA        <^X24>,0
                    0224  9679          VBUSDATA        <^X25>,0
                02  0225  9680          .BYTE   ^X2                             ;CLRQ
                    0226  9681          VBUSDATA        <^X24>,0
                    0227  9682          VBUSDATA        <^X25>,0
                02  0228  9683          .BYTE   ^X2
                    0229  9684          VBUSDATA        <^X24>,0
                    022A  9685          VBUSDATA        <^X25>,0
                02  022B  9686          .BYTE   ^X2
                    022C  9687          VBUSDATA        <^X24>,0
                    022D  9688          VBUSDATA        <^X25>,0
                02  022E  9689          .BYTE   ^X2
                    022F  9690          VBUSDATA        <^X24>,0
```

```
                                  0230          VBUSDATA        <^X25>,0
                              02  0231          .BYTE   ^X2
                                  0232          VBUSDATA        <^X24>,0
                                  0233          VBUSDATA        <^X25>,0
                              02  0234          .BYTE   ^X2                                ;MOVQ
                                  0235          VBUSDATA        <^X24>,1
                                  0236          VBUSDATA        <^X25>,1
                              02  0237          .BYTE   ^X2
                                  0238          VBUSDATA        <^X24>,0
                                  0239          VBUSDATA        <^X25>,0
                              02  023A          .BYTE   ^X2
                                  023B          VBUSDATA        <^X24>,0
                                  023C          VBUSDATA        <^X25>,0
                              02  023D          .BYTE   ^X2
                                  023E          VBUSDATA        <^X24>,0
                                  023F          VBUSDATA        <^X25>,0
                              02  0240          .BYTE   ^X2
                                  0241          VBUSDATA        <^X24>,0
                                  0242          VBUSDATA        <^X25>,0
                                  0243
                                  0243
                                  0243
                                  0243  BEGIN_CPU_DATA
                                  0002
                                  0002  DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 7700,4800,5BE4,0300,2470,1802  000C  ;% 01   WCTRL/LOADCRAR,WB_R[TEMP0]                ;DISABLE ^P INTERRUPT
U 02, 7700,0800,5BE4,0304,2C70,1803  0017  ;% 02   WCTRL/CONWRITE,WB_R[TEMP1]                ;...
U 03, 7700,9800,D370,030F,FA70,1804  0022  ;% 03   IPL_[1F]                                   ;SET IPL TO 1F
U 04, 5700,4840,0364,0308,0470,1805  002D  ;% 04   R[TEMP2]_RDM                               ;MOVE LOOP COUNT TO RTEMP2
U 05, 7700,D800,EF64,0301,0470,1806  0038  ;% 05   PL_[2]                                     ;SET PLATCH TO 2
U 06, 7700,C800,8B71,0308,0470,1807  0043  ;% 06   Q_R[TEMP2].RL.P                            ;ROTATE LOOP COUNT LEFT 2
U 07, 7700,4800,03A4,03D8,4870,1808  004E  ;% 07   WCTRL/PC_WB,WB_Q                           ;FORCE PREFETCH TO HAPPEN
U 08, 7700,4800,0364,1700,0470,1809  0059  ;% 08   IRD1TEST                                   ;LOAD IR REGISTER
U 09, 7600,4800,0364,0300,0470,180A  0064  ;% 09   DTYPE/IDEP,STROBE.V.BUS                    ;LOAD DSIZE INTO VBUS
U 0A, 7300,C800,0364,0300,0470,180B  006F  ;% 0A   NOP,STOP.CLOCK                             ;STOP THE CPU CLOCK
U 0B, 7700,4800,0364,0300,0470,180C  007A  ;% 0B   NOP
U 0C, 7700,C800,0364,1B00,0470,180D  0085  ;% 0C   LOD INC BRA?                               ;INCREMENT THE IRD COUNTER
U 0D, 7600,C800,0364,0300,0470,180E  0090  ;% 0D   DTYPE/IDEP,STROBE.V.BUS                    ;LOAD DSIZE INTO VBUS
U 0E, 7300,4800,0364,0300,0470,180F  009B  ;% 0E   NOP,STOP.CLOCK                             ;STOP THE CPU CLOCK
                                  00A6
                                  00A6  RTEMP_DATA
                                  0001
                          00C00000  0001          .LONG   ^X00C00000                         ;DISABLE ^P INTERRUPT
                          00400000  0005          .LONG   ^X00400000                         ;...
                                  0009
                                  0009  CACHE_DATA
                                  0001
                          00000065  0001          .LONG   ^X00000065                         ;MULD3
                          00000066  0005          .LONG   ^X00000066                         ;DIVD2
                          00000067  0009          .LONG   ^X00000067                         ;DIVD3
                          00000068  000D          .LONG   ^X00000068                         ;CVTDB
                          00000069  0011          .LONG   ^X00000069                         ;CVTDW
                          0000006A  0015          .LONG   ^X0000006A                         ;CVTDL
```

```
0000006B  0019  9791          .LONG   ^X0000006B                      ;CVTRDL
0000006C  001D  9792          .LONG   ^X0000006C                      ;CVTBD
0000006D  0021  9793          .LONG   ^X0000006D                      ;CVTWD
0000006E  0025  9794          .LONG   ^X0000006E                      ;CVTLD
0000006F  0029  9795          .LONG   ^X0000006F                      ;ACBD
00000070  002D  9796          .LONG   ^X00000070                      ;MOVD
00000071  0031  9797          .LONG   ^X00000071                      ;CMPD
00000072  0035  9798          .LONG   ^X00000072                      ;MNEGD
00000073  0039  9799          .LONG   ^X00000073                      ;TSTD
00000074  003D  9800          .LONG   ^X00000074                      ;EMODD
00000075  0041  9801          .LONG   ^X00000075                      ;POLYD
00000076  0045  9802          .LONG   ^X00000076                      ;CVTDF
00000077  0049  9803          .LONG   ^X00000077                      ;RESVRD
00000078  004D  9804          .LONG   ^X00000078                      ;ASHL
00000079  0051  9805          .LONG   ^X00000079                      ;ASHQ
0000007A  0055  9806          .LONG   ^X0000007A                      ;EMUL
0000007B  0059  9807          .LONG   ^X0000007B                      ;EDIV
0000007C  005D  9808          .LONG   ^X0000007C                      ;CLRQ
0000007D  0061  9809          .LONG   ^X0000007D                      ;MOVQ
00000000  0065  9810          .LONG   ^X00000000
          0069  9811
          0069  9812 END_CPU_DATA
          0243  9813
          0243  9814
          0243  9815 END_TEST_DATA
          0243
          0243          ;-----------------------------------------------------------------------------
          0243  9816 ENDTEST
```

```
                        0021        .SBTTL  "TEST   089     NATIVE MODE DSIZE ROM TEST 7
                        0021  9818 ;*********************************************************************************
                        0021  9819 ;++
                        0021  9820 ;
                        0021  9821 ; FUNCTIONAL DESCRIPTION:
                        0021  9822 ;
                        0021  9823 ;       THIS IS ONE IN A SERIES OF 14 TESTS THAT CHECK THE CONTENTS OF THE
                        0021  9824 ;       DSIZE ROM.
                        0021  9825 ;
                        0021  9826 ;
                        0021  9827 ; TEST ALGORITHM:
                        0021  9828 ;
                        0021  9829 ;       1.  CREATE AN INDEX (K) TO INDEX RESULTS TABLE
                        0021  9830 ;       2.  CREATE LOOP (I) OF 25
                        0021  9831 ;       3.  SAVE LOOP COUNT IN 1$
                        0021  9832 ;       4.  MOVE LOOP COUNT TO WDREG
                        0021  9833 ;       5.  EXECUTE DCS PROGRAM
                        0021  9834 ;               a.  DISABLE ^P INTERRUPTS
                        0021  9835 ;               b.  SET IPL TO 1F
                        0021  9836 ;               c.  MOVE LOOP COUNT TO RTEMP2
                        0021  9837 ;               d.  ROTATE LOOP COUNT LEFT 2 TO FORM CACHE ADDRESS
                        0021  9838 ;               e.  LOAD THIS ADDRESS INTO PC TO CAUSE PREFETCH
                        0021  9839 ;               f.  PERFORM IRD1TEST TO CAUSE IR TO LOAD INSTRUCTION
                        0021  9840 ;               g.  SET DTYPE TO INSTURCTION DEPENDENT AND STROBE THE VBUS
                        0021  9841 ;       6.  TEST VBUS FOR CORRECT DSIZE
                        0021  9842 ;       7.  IF NO ERROR CREATE LOOP (J) OF 5
                        0021  9843 ;       8.  EXECUTE DCS PROGRAM
                        0021  9844 ;               a.  INCREMENT THE IRD COUNTER
                        0021  9845 ;               b.  SET DTYPE TO INSTRUCTION DEPENDENT AND STROBE THE VBUS
                        0021  9846 ;       9.  TEST VBUS FOR CORRECT DSIZE
                        0021  9847 ;      10.  INCREMENT INDEX K BY ONE
                        0021  9848 ;      11.  IF NO ERROR GO TO STEP 8 UNTIL LOOP J IS EXHAUSTED
                        0021  9849 ;      12.  GO TO STEP 3 UNTIL LOOP I IS EXHAUSTED
                        0021  9850 ;
                        0021  9851 ;
                        0021  9852 ; TEST PATTERNS:
                        0021  9853 ;
                        0021  9854 ;       LOOP COUNT I            INSTRUCTION
                        0021  9855 ;            1                     MOVAQ
                        0021  9856 ;            2                     PUSHAQ
                        0021  9857 ;            3                     ADDB2
                        0021  9858 ;            4                     ADDB3
                        0021  9859 ;            5                     SUBB2
                        0021  9860 ;            6                     SUBB3
                        0021  9861 ;            7                     MULB2
                        0021  9862 ;            8                     MULB3
                        0021  9863 ;            9                     DIVB2
                        0021  9864 ;            A                     DIVB3
                        0021  9865 ;            B                     BISB2
                        0021  9866 ;            C                     BISB3
                        0021  9867 ;            D                     BICB2
                        0021  9868 ;            E                     BICB3
                        0021  9869 ;            F                     XORB2
                        0021  9870 ;           10                     XORB3
                        0021  9871 ;           11                     MNEGB
```

```
0021  ;          12                    CASEB
0021  ;          13                    MOVB
0021  ;          14                    CMPB
0021  ;          15                    MCOMB
0021  ;          16                    BITB
0021  ;          17                    CLRB
0021  ;          18                    TSTB
0021  ;          19                    INCB
0021  ;
0021  ;
0021  ; LOGIC DESCRIPTION:
0021  ;
0021  ;       THIS TEST ALONG WITH THE OTHER 13 TESTS IN THIS SERIES CHECK ALL THE
0021  ;       LOCATIONS BLASTED INTO THE DSIZE ROM.  ANY FALIURES SHOULD BE CONFINED
0021  ;       TO THE ROM.
0021  ;
0021  ;
0021  ; ASSUMPTIONS:
0021  ;
0021  ;       THIS TEST ASSUMES THE PREFETCH LOGIC AND INSTRUCTION REGISTER ARE
0021  ;       OPERATIONAL.
0021  ;
0021  ;
0021  ; ERROR DESCRIPTION:
0021  ;
0021  ;       EXPECTED VBUS DATA (DSIZE)
0021  ;       RECEIVED VBUS DATA
0021  ;       LOOP COUNT I
0021  ;       LOOP COUNT J
0021  ;       LOOP COUNT K
0021  ;               (BITS <7:0> = VBUS BIT POSITION)
0021  ;               (BITS <8> = BIT STATE)
0021  ;
0021  ;--
0021  ;*****************************************************************************
0021
0021
0021          INITIALIZE                                  ;INIT THE CPU
0023          LDINDEX         K,1                         ;INIT INDEX K TO 1
002A          LOOP            I,1,25                      ;CREATE A TEST LOOP OF 50
0031          SAVEINDEX       I,1$                        ;SAVE LOOP COUNT IN 1$
0036          LOADREG         WDREG,1$,,L                 ;LOAD LOOP COUNT INTO WDREG
003E          ERRLOOP                                     ;ERROR LOOP POINT
0040          FETCH           0                           ;START DCS PROGRAM
0045          BRSTCLK         <^X0A>,INH                  ;END DCS PROGRAM
0049          CMPVBUS         2$,I                        ;TEST DSIZE
004E          IFERROR                                     ;ROM FAILURE
0053          LOOP            J,1,5                       ;CREATE TEST LOOP OF 5
005A          FETCH           <^X0B>                      ;START DCS PROGRAM
005F          BRSTCLK         <^X0E>,INH                  ;END DCS PROGRAM
0063          CMPVBUS         3$,K                        ;TEST DSIZE
0068          IFERROR                                     ;ROM FAILURE
006D          INCINDEX        K                           ;INCREMENT INDEX K BY ONE
0070          ENDLOOP         J                           ;END TEST LOOP J
0073          ENDLOOP         I                           ;END TEST LOOP I
```

```
                    0076  9927          SKIP                                    ;GO TO NEXT TEST
                    007D  9928
                    007D  9929
                    007D  9930 BEGIN_TEST_DATA
                    007D
                    007D        ;-----------------------------------------------------------------------
                    007D  9931
          00000000  007D  9932 1$:      .LONG   ^X00000000                      ;LOOP COUNT STORAGE
                 02 0081  9933 2$:      .BYTE   ^X2                             ;RESULTS TABLE (IRD 0)
                    0082  9934          VBUSDATA        <^X24>,1                ;MOVAQ
                    0083  9935          VBUSDATA        <^X25>,1
                 02 0084  9936          .BYTE   ^X2                             ;PUSHAQ
                    0085  9937          VBUSDATA        <^X24>,1
                    0086  9938          VBUSDATA        <^X25>,1
                 02 0087  9939          .BYTE   ^X2                             ;ADDB2
                    0088  9940          VBUSDATA        <^X24>,0
                    0089  9941          VBUSDATA        <^X25>,0
                 02 008A  9942          .BYTE   ^X2                             ;ADDB3
                    008B  9943          VBUSDATA        <^X24>,0
                    008C  9944          VBUSDATA        <^X25>,0
                 02 008D  9945          .BYTE   ^X2                             ;SUBB2
                    008E  9946          VBUSDATA        <^X24>,0
                    008F  9947          VBUSDATA        <^X25>,0
                 02 0090  9948          .BYTE   ^X2                             ;SUBB3
                    0091  9949          VBUSDATA        <^X24>,0
                    0092  9950          VBUSDATA        <^X25>,0
                 02 0093  9951          .BYTE   ^X2                             ;MULB2
                    0094  9952          VBUSDATA        <^X24>,0
                    0095  9953          VBUSDATA        <^X25>,0
                 02 0096  9954          .BYTE   ^X2                             ;MULB3
                    0097  9955          VBUSDATA        <^X24>,0
                    0098  9956          VBUSDATA        <^X25>,0
                 02 0099  9957          .BYTE   ^X2                             ;DIVB2
                    009A  9958          VBUSDATA        <^X24>,0
                    009B  9959          VBUSDATA        <^X25>,0
                 02 009C  9960          .BYTE   ^X2                             ;DIVB3
                    009D  9961          VBUSDATA        <^X24>,0
                    009E  9962          VBUSDATA        <^X25>,0
                 02 009F  9963          .BYTE   ^X2                             ;BISB2
                    00A0  9964          VBUSDATA        <^X24>,0
                    00A1  9965          VBUSDATA        <^X25>,0
                 02 00A2  9966          .BYTE   ^X2                             ;BISB3
                    00A3  9967          VBUSDATA        <^X24>,0
                    00A4  9968          VBUSDATA        <^X25>,0
                 02 00A5  9969          .BYTE   ^X2                             ;BICB2
                    00A6  9970          VBUSDATA        <^X24>,0
                    00A7  9971          VBUSDATA        <^X25>,0
                 02 00A8  9972          .BYTE   ^X2                             ;BICB3
                    00A9  9973          VBUSDATA        <^X24>,0
                    00AA  9974          VBUSDATA        <^X25>,0
                 02 00AB  9975          .BYTE   ^X2                             ;XORB2
                    00AC  9976          VBUSDATA        <^X24>,0
                    00AD  9977          VBUSDATA        <^X25>,0
                 02 00AE  9978          .BYTE   ^X2                             ;XORB3
                    00AF  9979          VBUSDATA        <^X24>,0
```

```
                    00B0  9980          VBUSDATA        <^X25>,0
                02  00B1  9981          .BYTE   ^X2                             ;MNEGB
                    00B2  9982          VBUSDATA        <^X24>,0
                    00B3  9983          VBUSDATA        <^X25>,0
                02  00B4  9984          .BYTE   ^X2                             ;CASEB
                    00B5  9985          VBUSDATA        <^X24>,0
                    00B6  9986          VBUSDATA        <^X25>,0
                02  00B7  9987          .BYTE   ^X2                             ;MOVB
                    00B8  9988          VBUSDATA        <^X24>,0
                    00B9  9989          VBUSDATA        <^X25>,0
                02  00BA  9990          .BYTE   ^X2                             ;CMPB
                    00BB  9991          VBUSDATA        <^X24>,0
                    00BC  9992          VBUSDATA        <^X25>,0
                02  00BD  9993          .BYTE   ^X2                             ;MCOMB
                    00BE  9994          VBUSDATA        <^X24>,0
                    00BF  9995          VBUSDATA        <^X25>,0
                02  00C0  9996          .BYTE   ^X2                             ;BITB
                    00C1  9997          VBUSDATA        <^X24>,0
                    00C2  9998          VBUSDATA        <^X25>,0
                02  00C3  9999          .BYTE   ^X2                             ;CLRB
                    00C4 10000          VBUSDATA        <^X24>,0
                    00C5 10001          VBUSDATA        <^X25>,0
                02  00C6 10002          .BYTE   ^X2                             ;TSTB
                    00C7 10003          VBUSDATA        <^X24>,0
                    00C8 10004          VBUSDATA        <^X25>,0
                02  00C9 10005          .BYTE   ^X2                             ;INCB
                    00CA 10006          VBUSDATA        <^X24>,0
                    00CB 10007          VBUSDATA        <^X25>,0
                    00CC 10008
                02  00CC 10009 3$:      .BYTE   ^X2                             ;RESULTS TABLE (IRD 1 TO 5)
                    00CD 10010          VBUSDATA        <^X24>,0                ;MOVAQ
                    00CE 10011          VBUSDATA        <^X25>,1
                02  00CF 10012          .BYTE   ^X2
                    00D0 10013          VBUSDATA        <^X24>,0
                    00D1 10014          VBUSDATA        <^X25>,0
                02  00D2 10015          .BYTE   ^X2
                    00D3 10016          VBUSDATA        <^X24>,0
                    00D4 10017          VBUSDATA        <^X25>,0
                02  00D5 10018          .BYTE   ^X2
                    00D6 10019          VBUSDATA        <^X24>,0
                    00D7 10020          VBUSDATA        <^X25>,0
                02  00D8 10021          .BYTE   ^X2
                    00D9 10022          VBUSDATA        <^X24>,0
                    00DA 10023          VBUSDATA        <^X25>,0
                02  00DB 10024          .BYTE   ^X2                             ;PUSHAQ
                    00DC 10025          VBUSDATA        <^X24>,0
                    00DD 10026          VBUSDATA        <^X25>,0
                02  00DE 10027          .BYTE   ^X2
                    00DF 10028          VBUSDATA        <^X24>,0
                    00E0 10029          VBUSDATA        <^X25>,0
                02  00E1 10030          .BYTE   ^X2
                    00E2 10031          VBUSDATA        <^X24>,0
                    00E3 10032          VBUSDATA        <^X25>,0
                02  00E4 10033          .BYTE   ^X2
                    00E5 10034          VBUSDATA        <^X24>,0
```

```
                    00E6 10035          VBUSDATA        <^X25>,0
                02  00E7 10036          .BYTE   ^X2
                    00E8 10037          VBUSDATA        <^X24>,0
                    00E9 10038          VBUSDATA        <^X25>,0
                02  00EA 10039          .BYTE   ^X2                             ;ADDB2
                    00EB 10040          VBUSDATA        <^X24>,0
                    00EC 10041          VBUSDATA        <^X25>,0
                02  00ED 10042          .BYTE   ^X2
                    00EE 10043          VBUSDATA        <^X24>,0
                    00EF 10044          VBUSDATA        <^X25>,0
                02  00F0 10045          .BYTE   ^X2
                    00F1 10046          VBUSDATA        <^X24>,0
                    00F2 10047          VBUSDATA        <^X25>,0
                02  00F3 10048          .BYTE   ^X2
                    00F4 10049          VBUSDATA        <^X24>,0
                    00F5 10050          VBUSDATA        <^X25>,0
                02  00F6 10051          .BYTE   ^X2
                    00F7 10052          VBUSDATA        <^X24>,0
                    00F8 10053          VBUSDATA        <^X25>,0
                02  00F9 10054          .BYTE   ^X2                             ;ADDB3
                    00FA 10055          VBUSDATA        <^X24>,0
                    00FB 10056          VBUSDATA        <^X25>,0
                02  00FC 10057          .BYTE   ^X2
                    00FD 10058          VBUSDATA        <^X24>,0
                    00FE 10059          VBUSDATA        <^X25>,0
                02  00FF 10060          .BYTE   ^X2
                    0100 10061          VBUSDATA        <^X24>,0
                    0101 10062          VBUSDATA        <^X25>,0
                02  0102 10063          .BYTE   ^X2
                    0103 10064          VBUSDATA        <^X24>,0
                    0104 10065          VBUSDATA        <^X25>,0
                02  0105 10066          .BYTE   ^X2
                    0106 10067          VBUSDATA        <^X24>,0
                    0107 10068          VBUSDATA        <^X25>,0
                02  0108 10069          .BYTE   ^X2                             ;SUBB2
                    0109 10070          VBUSDATA        <^X24>,0
                    010A 10071          VBUSDATA        <^X25>,0
                02  010B 10072          .BYTE   ^X2
                    010C 10073          VBUSDATA        <^X24>,0
                    010D 10074          VBUSDATA        <^X25>,0
                02  010E 10075          .BYTE   ^X2
                    010F 10076          VBUSDATA        <^X24>,0
                    0110 10077          VBUSDATA        <^X25>,0
                02  0111 10078          .BYTE   ^X2
                    0112 10079          VBUSDATA        <^X24>,0
                    0113 10080          VBUSDATA        <^X25>,0
                02  0114 10081          .BYTE   ^X2
                    0115 10082          VBUSDATA        <^X24>,0
                    0116 10083          VBUSDATA        <^X25>,0
                02  0117 10084          .BYTE   ^X2                             ;SUBB3
                    0118 10085          VBUSDATA        <^X24>,0
                    0119 10086          VBUSDATA        <^X25>,0
                02  011A 10087          .BYTE   ^X2
                    011B 10088          VBUSDATA        <^X24>,0
                    011C 10089          VBUSDATA        <^X25>,0
```

```
                    02  011D 10090          .BYTE   ^X2
                        011E 10091          VBUSDATA        <^X24>,0
                        011F 10092          VBUSDATA        <^X25>,0
                    02  0120 10093          .BYTE   ^X2
                        0121 10094          VBUSDATA        <^X24>,0
                        0122 10095          VBUSDATA        <^X25>,0
                    02  0123 10096          .BYTE   ^X2
                        0124 10097          VBUSDATA        <^X24>,0
                        0125 10098          VBUSDATA        <^X25>,0
                    02  0126 10099          .BYTE   ^X2                             ;MULB2
                        0127 10100          VBUSDATA        <^X24>,0
                        0128 10101          VBUSDATA        <^X25>,0
                    02  0129 10102          .BYTE   ^X2
                        012A 10103          VBUSDATA        <^X24>,0
                        012B 10104          VBUSDATA        <^X25>,0
                    02  012C 10105          .BYTE   ^X2
                        012D 10106          VBUSDATA        <^X24>,0
                        012E 10107          VBUSDATA        <^X25>,0
                    02  012F 10108          .BYTE   ^X2
                        0130 10109          VBUSDATA        <^X24>,0
                        0131 10110          VBUSDATA        <^X25>,0
                    02  0132 10111          .BYTE   ^X2
                        0133 10112          VBUSDATA        <^X24>,0
                        0134 10113          VBUSDATA        <^X25>,0
                    02  0135 10114          .BYTE   ^X2                             ;MULB3
                        0136 10115          VBUSDATA        <^X24>,0
                        0137 10116          VBUSDATA        <^X25>,0
                    02  0138 10117          .BYTE   ^X2
                        0139 10118          VBUSDATA        <^X24>,0
                        013A 10119          VBUSDATA        <^X25>,0
                    02  013B 10120          .BYTE   ^X2
                        013C 10121          VBUSDATA        <^X24>,0
                        013D 10122          VBUSDATA        <^X25>,0
                    02  013E 10123          .BYTE   ^X2
                        013F 10124          VBUSDATA        <^X24>,0
                        0140 10125          VBUSDATA        <^X25>,0
                    02  0141 10126          .BYTE   ^X2
                        0142 10127          VBUSDATA        <^X24>,0
                        0143 10128          VBUSDATA        <^X25>,0
                    02  0144 10129          .BYTE   ^X2                             ;DIVB2
                        0145 10130          VBUSDATA        <^X24>,0
                        0146 10131          VBUSDATA        <^X25>,0
                    02  0147 10132          .BYTE   ^X2
                        0148 10133          VBUSDATA        <^X24>,0
                        0149 10134          VBUSDATA        <^X25>,0
                    02  014A 10135          .BYTE   ^X2
                        014B 10136          VBUSDATA        <^X24>,0
                        014C 10137          VBUSDATA        <^X25>,0
                    02  014D 10138          .BYTE   ^X2
                        014E 10139          VBUSDATA        <^X24>,0
                        014F 10140          VBUSDATA        <^X25>,0
                    02  0150 10141          .BYTE   ^X2
                        0151 10142          VBUSDATA        <^X24>,0
                        0152 10143          VBUSDATA        <^X25>,0
                    02  0153 10144          .BYTE   ^X2                             ;DIVB3
```

```
                    0154 10145          VBUSDATA        <^X24>,0
                    0155 10146          VBUSDATA        <^X25>,0
                02  0156 10147          .BYTE   ^X2
                    0157 10148          VBUSDATA        <^X24>,0
                    0158 10149          VBUSDATA        <^X25>,0
                02  0159 10150          .BYTE   ^X2
                    015A 10151          VBUSDATA        <^X24>,0
                    015B 10152          VBUSDATA        <^X25>,0
                02  015C 10153          .BYTE   ^X2
                    015D 10154          VBUSDATA        <^X24>,0
                    015E 10155          VBUSDATA        <^X25>,0
                02  015F 10156          .BYTE   ^X2
                    0160 10157          VBUSDATA        <^X24>,0
                    0161 10158          VBUSDATA        <^X25>,0
                02  0162 10159          .BYTE   ^X2                             ;BISB2
                    0163 10160          VBUSDATA        <^X24>,0
                    0164 10161          VBUSDATA        <^X25>,0
                02  0165 10162          .BYTE   ^X2
                    0166 10163          VBUSDATA        <^X24>,0
                    0167 10164          VBUSDATA        <^X25>,0
                02  0168 10165          .BYTE   ^X2
                    0169 10166          VBUSDATA        <^X24>,0
                    016A 10167          VBUSDATA        <^X25>,0
                02  016B 10168          .BYTE   ^X2
                    016C 10169          VBUSDATA        <^X24>,0
                    016D 10170          VBUSDATA        <^X25>,0
                02  016E 10171          .BYTE   ^X2
                    016F 10172          VBUSDATA        <^X24>,0
                    0170 10173          VBUSDATA        <^X25>,0
                02  0171 10174          .BYTE   ^X2                         ;BISB3
                    0172 10175          VBUSDATA        <^X24>,0
                    0173 10176          VBUSDATA        <^X25>,0
                02  0174 10177          .BYTE   ^X2
                    0175 10178          VBUSDATA        <^X24>,0
                    0176 10179          VBUSDATA        <^X25>,0
                02  0177 10180          .BYTE   ^X2
                    0178 10181          VBUSDATA        <^X24>,0
                    0179 10182          VBUSDATA        <^X25>,0
                02  017A 10183          .BYTE   ^X2
                    017B 10184          VBUSDATA        <^X24>,0
                    017C 10185          VBUSDATA        <^X25>,0
                02  017D 10186          .BYTE   ^X2
                    017E 10187          VBUSDATA        <^X24>,0
                    017F 10188          VBUSDATA        <^X25>,0
                02  0180 10189          .BYTE   ^X2                         ;BICB2
                    0181 10190          VBUSDATA        <^X24>,0
                    0182 10191          VBUSDATA        <^X25>,0
                02  0183 10192          .BYTE   ^X2
                    0184 10193          VBUSDATA        <^X24>,0
                    0185 10194          VBUSDATA        <^X25>,0
                02  0186 10195          .BYTE   ^X2
                    0187 10196          VBUSDATA        <^X24>,0
                    0188 10197          VBUSDATA        <^X25>,0
                02  0189 10198          .BYTE   ^X2
                    018A 10199          VBUSDATA        <^X24>,0
```

```
                    018B 10200          VBUSDATA        <^X25>,0
                02  018C 10201          .BYTE   ^X2
                    018D 10202          VBUSDATA        <^X24>,0
                    018E 10203          VBUSDATA        <^X25>,0
                02  018F 10204          .BYTE   ^X2                             ;BICB3
                    0190 10205          VBUSDATA        <^X24>,0
                    0191 10206          VBUSDATA        <^X25>,0
                02  0192 10207          .BYTE   ^X2
                    0193 10208          VBUSDATA        <^X24>,0
                    0194 10209          VBUSDATA        <^X25>,0
                02  0195 10210          .BYTE   ^X2
                    0196 10211          VBUSDATA        <^X24>,0
                    0197 10212          VBUSDATA        <^X25>,0
                02  0198 10213          .BYTE   ^X2
                    0199 10214          VBUSDATA        <^X24>,0
                    019A 10215          VBUSDATA        <^X25>,0
                02  019B 10216          .BYTE   ^X2
                    019C 10217          VBUSDATA        <^X24>,0
                    019D 10218          VBUSDATA        <^X25>,0
                02  019E 10219          .BYTE   ^X2                             ;XORB2
                    019F 10220          VBUSDATA        <^X24>,0
                    01A0 10221          VBUSDATA        <^X25>,0
                02  01A1 10222          .BYTE   ^X2
                    01A2 10223          VBUSDATA        <^X24>,0
                    01A3 10224          VBUSDATA        <^X25>,0
                02  01A4 10225          .BYTE   ^X2
                    01A5 10226          VBUSDATA        <^X24>,0
                    01A6 10227          VBUSDATA        <^X25>,0
                02  01A7 10228          .BYTE   ^X2
                    01A8 10229          VBUSDATA        <^X24>,0
                    01A9 10230          VBUSDATA        <^X25>,0
                02  01AA 10231          .BYTE   ^X2
                    01AB 10232          VBUSDATA        <^X24>,0
                    01AC 10233          VBUSDATA        <^X25>,0
                02  01AD 10234          .BYTE   ^X2                         ;XORB3
                    01AE 10235          VBUSDATA        <^X24>,0
                    01AF 10236          VBUSDATA        <^X25>,0
                02  01B0 10237          .BYTE   ^X2
                    01B1 10238          VBUSDATA        <^X24>,0
                    01B2 10239          VBUSDATA        <^X25>,0
                02  01B3 10240          .BYTE   ^X2
                    01B4 10241          VBUSDATA        <^X24>,0
                    01B5 10242          VBUSDATA        <^X25>,0
                02  01B6 10243          .BYTE   ^X2
                    01B7 10244          VBUSDATA        <^X24>,0
                    01B8 10245          VBUSDATA        <^X25>,0
                02  01B9 10246          .BYTE   ^X2
                    01BA 10247          VBUSDATA        <^X24>,0
                    01BB 10248          VBUSDATA        <^X25>,0
                02  01BC 10249          .BYTE   ^X2                         ;MNEGB
                    01BD 10250          VBUSDATA        <^X24>,0
                    01BE 10251          VBUSDATA        <^X25>,0
                02  01BF 10252          .BYTE   ^X2
                    01C0 10253          VBUSDATA        <^X24>,0
                    01C1 10254          VBUSDATA        <^X25>,0
```

```
                02  01C2 10255          .BYTE   ^X2
                    01C3 10256          VBUSDATA        <^X24>,0
                    01C4 10257          VBUSDATA        <^X25>,0
                02  01C5 10258          .BYTE   ^X2
                    01C6 10259          VBUSDATA        <^X24>,0
                    01C7 10260          VBUSDATA        <^X25>,0
                02  01C8 10261          .BYTE   ^X2
                    01C9 10262          VBUSDATA        <^X24>,0
                    01CA 10263          VBUSDATA        <^X25>,0
                02  01CB 10264          .BYTE   ^X2                             ;CASEB
                    01CC 10265          VBUSDATA        <^X24>,0
                    01CD 10266          VBUSDATA        <^X25>,0
                02  01CE 10267          .BYTE   ^X2
                    01CF 10268          VBUSDATA        <^X24>,0
                    01D0 10269          VBUSDATA        <^X25>,0
                02  01D1 10270          .BYTE   ^X2
                    01D2 10271          VBUSDATA        <^X24>,0
                    01D3 10272          VBUSDATA        <^X25>,0
                02  01D4 10273          .BYTE   ^X2
                    01D5 10274          VBUSDATA        <^X24>,0
                    01D6 10275          VBUSDATA        <^X25>,0
                02  01D7 10276          .BYTE   ^X2
                    01D8 10277          VBUSDATA        <^X24>,0
                    01D9 10278          VBUSDATA        <^X25>,0
                02  01DA 10279          .BYTE   ^X2                             ;MOVB
                    01DB 10280          VBUSDATA        <^X24>,0
                    01DC 10281          VBUSDATA        <^X25>,0
                02  01DD 10282          .BYTE   ^X2
                    01DE 10283          VBUSDATA        <^X24>,0
                    01DF 10284          VBUSDATA        <^X25>,0
                02  01E0 10285          .BYTE   ^X2
                    01E1 10286          VBUSDATA        <^X24>,0
                    01E2 10287          VBUSDATA        <^X25>,0
                02  01E3 10288          .BYTE   ^X2
                    01E4 10289          VBUSDATA        <^X24>,0
                    01E5 10290          VBUSDATA        <^X25>,0
                02  01E6 10291          .BYTE   ^X2
                    01E7 10292          VBUSDATA        <^X24>,0
                    01E8 10293          VBUSDATA        <^X25>,0
                02  01E9 10294          .BYTE   ^X2                             ;CMPB
                    01EA 10295          VBUSDATA        <^X24>,0
                    01EB 10296          VBUSDATA        <^X25>,0
                02  01EC 10297          .BYTE   ^X2
                    01ED 10298          VBUSDATA        <^X24>,0
                    01EE 10299          VBUSDATA        <^X25>,0
                02  01EF 10300          .BYTE   ^X2
                    01F0 10301          VBUSDATA        <^X24>,0
                    01F1 10302          VBUSDATA        <^X25>,0
                02  01F2 10303          .BYTE   ^X2
                    01F3 10304          VBUSDATA        <^X24>,0
                    01F4 10305          VBUSDATA        <^X25>,0
                02  01F5 10306          .BYTE   ^X2
                    01F6 10307          VBUSDATA        <^X24>,0
                    01F7 10308          VBUSDATA        <^X25>,0
                02  01F8 10309          .BYTE   ^X2                             ;MCOMB
```

```
                    01F9 10310          VBUSDATA        <^X24>,0
                    01FA 10311          VBUSDATA        <^X25>,0
                02  01FB 10312          .BYTE   ^X2
                    01FC 10313          VBUSDATA        <^X24>,0
                    01FD 10314          VBUSDATA        <^X25>,0
                02  01FE 10315          .BYTE   ^X2
                    01FF 10316          VBUSDATA        <^X24>,0
                    0200 10317          VBUSDATA        <^X25>,0
                02  0201 10318          .BYTE   ^X2
                    0202 10319          VBUSDATA        <^X24>,0
                    0203 10320          VBUSDATA        <^X25>,0
                02  0204 10321          .BYTE   ^X2
                    0205 10322          VBUSDATA        <^X24>,0
                    0206 10323          VBUSDATA        <^X25>,0
                02  0207 10324          .BYTE   ^X2                             ;BITB
                    0208 10325          VBUSDATA        <^X24>,0
                    0209 10326          VBUSDATA        <^X25>,0
                02  020A 10327          .BYTE   ^X2
                    020B 10328          VBUSDATA        <^X24>,0
                    020C 10329          VBUSDATA        <^X25>,0
                02  020D 10330          .BYTE   ^X2
                    020E 10331          VBUSDATA        <^X24>,0
                    020F 10332          VBUSDATA        <^X25>,0
                02  0210 10333          .BYTE   ^X2
                    0211 10334          VBUSDATA        <^X24>,0
                    0212 10335          VBUSDATA        <^X25>,0
                02  0213 10336          .BYTE   ^X2
                    0214 10337          VBUSDATA        <^X24>,0
                    0215 10338          VBUSDATA        <^X25>,0
                02  0216 10339          .BYTE   ^X2                             ;CLRB
                    0217 10340          VBUSDATA        <^X24>,0
                    0218 10341          VBUSDATA        <^X25>,0
                02  0219 10342          .BYTE   ^X2
                    021A 10343          VBUSDATA        <^X24>,0
                    021B 10344          VBUSDATA        <^X25>,0
                02  021C 10345          .BYTE   ^X2
                    021D 10346          VBUSDATA        <^X24>,0
                    021E 10347          VBUSDATA        <^X25>,0
                02  021F 10348          .BYTE   ^X2
                    0220 10349          VBUSDATA        <^X24>,0
                    0221 10350          VBUSDATA        <^X25>,0
                02  0222 10351          .BYTE   ^X2
                    0223 10352          VBUSDATA        <^X24>,0
                    0224 10353          VBUSDATA        <^X25>,0
                02  0225 10354          .BYTE   ^X2                             ;TSTB
                    0226 10355          VBUSDATA        <^X24>,0
                    0227 10356          VBUSDATA        <^X25>,0
                02  0228 10357          .BYTE   ^X2
                    0229 10358          VBUSDATA        <^X24>,0
                    022A 10359          VBUSDATA        <^X25>,0
                02  022B 10360          .BYTE   ^X2
                    022C 10361          VBUSDATA        <^X24>,0
                    022D 10362          VBUSDATA        <^X25>,0
                02  022E 10363          .BYTE   ^X2
                    022F 10364          VBUSDATA        <^X24>,0
```

```
                                0230 10365          VBUSDATA        <^X25>,0
                          02    0231 10366          .BYTE   ^X2
                                0232 10367          VBUSDATA        <^X24>,0
                                0233 10368          VBUSDATA        <^X25>,0
                          02    0234 10369          .BYTE   ^X2                             ;INCB
                                0235 10370          VBUSDATA        <^X24>,0
                                0236 10371          VBUSDATA        <^X25>,0
                          02    0237 10372          .BYTE   ^X2
                                0238 10373          VBUSDATA        <^X24>,0
                                0239 10374          VBUSDATA        <^X25>,0
                          02    023A 10375          .BYTE   ^X2
                                023B 10376          VBUSDATA        <^X24>,0
                                023C 10377          VBUSDATA        <^X25>,0
                          02    023D 10378          .BYTE   ^X2
                                023E 10379          VBUSDATA        <^X24>,0
                                023F 10380          VBUSDATA        <^X25>,0
                          02    0240 10381          .BYTE   ^X2
                                0241 10382          VBUSDATA        <^X24>,0
                                0242 10383          VBUSDATA        <^X25>,0
                                0243 10384
                                0243 10385
                                0243 10386
                                0243 10387 BEGIN_CPU_DATA
                                0002 10388
                                0002 10389 DCS_DATA
                                0001 10390
U 00, 7700,C800,0364,0300,0470,1801   0001 10391 ;% 00    NOP
U 01, 7700,4800,5BE4,0300,2470,1802   000C 10395 ;% 01    WCTRL/LOADCRAR,WB_R[TEMP0]          ;DISABLE ^P INTERRUPT
U 02, 7700,0800,5BE4,0304,2C70,1803   0017 10399 ;% 02    WCTRL/CONWRITE,WB_R[TEMP1]          ;...
U 03, 7700,9800,D370,030F,FA70,1804   0022 10403 ;% 03    IPL_[1F]                            ;SET IPL TO 1F
U 04, 5700,4840,0364,0308,0470,1805   002D 10407 ;% 04    R[TEMP2]_RDM                        ;MOVE LOOP COUNT TO RTEMP2
U 05, 7700,D800,EF64,0301,0470,1806   0038 10411 ;% 05    PL_[2]                              ;SET PLATCH TO 2
U 06, 7700,C800,8B71,0308,0470,1807   0043 10415 ;% 06    Q_R[TEMP2].RL.P                     ;ROTATE LOOP COUNT LEFT 2
U 07, 7700,4800,03A4,03D8,4870,1808   004E 10419 ;% 07    WCTRL/PC_WB,WB_Q                    ;FORCE PREFETCH TO HAPPEN
U 08, 7700,4800,0364,1700,0470,1809   0059 10423 ;% 08    IRD1TEST                            ;LOAD IR REGISTER
U 09, 7600,4800,0364,0300,0470,180A   0064 10427 ;% 09    DTYPE/IDEP,STROBE.V.BUS             ;LOAD DSIZE INTO VBUS
U 0A, 7300,C800,0364,0300,0470,180B   006F 10431 ;% 0A    NOP,STOP.CLOCK                      ;STOP THE CPU CLOCK
U 0B, 7700,4800,0364,0300,0470,180C   007A 10435 ;% 0B    NOP
U 0C, 7700,C800,0364,1B00,0470,180D   0085 10439 ;% 0C    LOD INC BRA?                        ;INCREMENT THE IRD COUNTER
U 0D, 7600,C800,0364,0300,0470,180E   0090 10443 ;% 0D    DTYPE/IDEP,STROBE.V.BUS             ;LOAD DSIZE INTO VBUS
U 0E, 7300,4800,0364,0300,0470,180F   009B 10447 ;% 0E    NOP,STOP.CLOCK                      ;STOP THE CPU CLOCK
                                00A6 10451
                                00A6 10452 RTEMP_DATA
                                0001 10453
                          00C00000    0001 10454          .LONG   ^X00C00000                  ;DISABLE ^P INTERRUPT
                          00400000    0005 10455          .LONG   ^X00400000                  ;...
                                0009 10456
                                0009 10457 CACHE_DATA
                                0001 10458
                          0000007E    0001 10459          .LONG   ^X0000007E                  ;MOVAQ
                          0000007F    0005 10460          .LONG   ^X0000007F                  ;PUSHAQ
                          00000080    0009 10461          .LONG   ^X00000080                  ;ADDB2
                          00000081    000D 10462          .LONG   ^X00000081                  ;ADDB3
                          00000082    0011 10463          .LONG   ^X00000082                  ;SUBB2
                          00000083    0015 10464          .LONG   ^X00000083                  ;SUBB3
```

```
00000084  0019 10465          .LONG   ^X00000084                      ;MULB2
00000085  001D 10466          .LONG   ^X00000085                      ;MULB3
00000086  0021 10467          .LONG   ^X00000086                      ;DIVB2
00000087  0025 10468          .LONG   ^X00000087                      ;DIVB3
00000088  0029 10469          .LONG   ^X00000088                      ;BISB2
00000089  002D 10470          .LONG   ^X00000089                      ;BISB3
0000008A  0031 10471          .LONG   ^X0000008A                      ;BICB2
0000008B  0035 10472          .LONG   ^X0000008B                      ;BICB3
0000008C  0039 10473          .LONG   ^X0000008C                      ;XORB2
0000008D  003D 10474          .LONG   ^X0000008D                      ;XORB3
0000008E  0041 10475          .LONG   ^X0000008E                      ;MNEGB
0000008F  0045 10476          .LONG   ^X0000008F                      ;CASEB
00000090  0049 10477          .LONG   ^X00000090                      ;MOVB
00000091  004D 10478          .LONG   ^X00000091                      ;CMPB
00000092  0051 10479          .LONG   ^X00000092                      ;MCOMB
00000093  0055 10480          .LONG   ^X00000093                      ;BITB
00000094  0059 10481          .LONG   ^X00000094                      ;CLRB
00000095  005D 10482          .LONG   ^X00000095                      ;TSTB
00000096  0061 10483          .LONG   ^X00000096                      ;INCB
00000000  0065 10484          .LONG   ^X00000000
          0069 10485
          0069 10486 END_CPU_DATA
          0243 10487
          0243 10488
          0243 10489 END_TEST_DATA
          0243
          0243        ;-----------------------------------------------------------------------------
          0243 10490 ENDTEST
```

```
                        0021        .SBTTL  "TEST   08A     NATIVE MODE DSIZE ROM TEST 8
                        0021 10492 ;*************************************************************************
                        0021 10493 ;++
                        0021 10494 ;
                        0021 10495 ; FUNCTIONAL DESCRIPTION:
                        0021 10496 ;
                        0021 10497 ;       THIS IS ONE IN A SERIES OF 14 TESTS THAT CHECK THE CONTENTS OF THE
                        0021 10498 ;       DSIZE ROM.
                        0021 10499 ;
                        0021 10500 ;
                        0021 10501 ; TEST ALGORITHM:
                        0021 10502 ;
                        0021 10503 ;       1.  CREATE AN INDEX (K) TO INDEX RESULTS TABLE
                        0021 10504 ;       2.  CREATE LOOP (I) OF 25
                        0021 10505 ;       3.  SAVE LOOP COUNT IN 1$
                        0021 10506 ;       4.  MOVE LOOP COUNT TO WDREG
                        0021 10507 ;       5.  EXECUTE DCS PROGRAM
                        0021 10508 ;               a.  DISABLE ^P INTERRUPTS
                        0021 10509 ;               b.  SET IPL TO 1F
                        0021 10510 ;               c.  MOVE LOOP COUNT TO RTEMP2
                        0021 10511 ;               d.  ROTATE LOOP COUNT LEFT 2 TO FORM CACHE ADDRESS
                        0021 10512 ;               e.  LOAD THIS ADDRESS INTO PC TO CAUSE PREFETCH
                        0021 10513 ;               f.  PERFORM IRD1TEST TO CAUSE IR TO LOAD INSTRUCTION
                        0021 10514 ;               g.  SET DTYPE TO INSTURCTION DEPENDENT AND STROBE THE VBUS
                        0021 10515 ;       6.  TEST VBUS FOR CORRECT DSIZE
                        0021 10516 ;       7.  IF NO ERROR CREATE LOOP (J) OF 5
                        0021 10517 ;       8.  EXECUTE DCS PROGRAM
                        0021 10518 ;               a.  INCREMENT THE IRD COUNTER
                        0021 10519 ;               b.  SET DTYPE TO INSTRUCTION DEPENDENT AND STROBE THE VBUS
                        0021 10520 ;       9.  TEST VBUS FOR CORRECT DSIZE
                        0021 10521 ;      10.  INCREMENT INDEX K BY ONE
                        0021 10522 ;      11.  IF NO ERROR GO TO STEP 8 UNTIL LOOP J IS EXHAUSTED
                        0021 10523 ;      12.  GO TO STEP 3 UNTIL LOOP I IS EXHAUSTED
                        0021 10524 ;
                        0021 10525 ;
                        0021 10526 ; TEST PATTERNS:
                        0021 10527 ;
                        0021 10528 ;       LOOP COUNT I             INSTRUCTION
                        0021 10529 ;            1                      DECB
                        0021 10530 ;            2                      CVTBL
                        0021 10531 ;            3                      CVTBW
                        0021 10532 ;            4                      MOVZBL
                        0021 10533 ;            5                      MOVZBW
                        0021 10534 ;            6                      ROTL
                        0021 10535 ;            7                      ACBB
                        0021 10536 ;            8                      MOVAB
                        0021 10537 ;            9                      PUSHAB
                        0021 10538 ;            A                      ADDW2
                        0021 10539 ;            B                      ADDW3
                        0021 10540 ;            C                      SUBW2
                        0021 10541 ;            D                      SUBW3
                        0021 10542 ;            E                      MULW2
                        0021 10543 ;            F                      MULW3
                        0021 10544 ;           10                      DIVW2
                        0021 10545 ;           11                      DIVW3
```

```
0021 10546 ;          12                    BISW2
0021 10547 ;          13                    BISW3
0021 10548 ;          14                    BICW2
0021 10549 ;          15                    BICW3
0021 10550 ;          16                    XORW2
0021 10551 ;          17                    XORW3
0021 10552 ;          18                    MNEGW
0021 10553 ;          19                    CASEW
0021 10554 ;
0021 10555 ;
0021 10556 ; LOGIC DESCRIPTION:
0021 10557 ;
0021 10558 ;       THIS TEST ALONG WITH THE OTHER 13 TESTS IN THIS SERIES CHECK ALL THE
0021 10559 ;       LOCATIONS BLASTED INTO THE DSIZE ROM.  ANY FALIURES SHOULD BE CONFINED
0021 10560 ;       TO THE ROM.
0021 10561 ;
0021 10562 ;
0021 10563 ; ASSUMPTIONS:
0021 10564 ;
0021 10565 ;       THIS TEST ASSUMES THE PREFETCH LOGIC AND INSTRUCTION REGISTER ARE
0021 10566 ;       OPERATIONAL.
0021 10567 ;
0021 10568 ;
0021 10569 ; ERROR DESCRIPTION:
0021 10570 ;
0021 10571 ;       EXPECTED VBUS DATA (DSIZE)
0021 10572 ;       RECEIVED VBUS DATA
0021 10573 ;       LOOP COUNT I
0021 10574 ;       LOOP COUNT J
0021 10575 ;       LOOP COUNT K
0021 10576 ;               (BITS <7:0> = VBUS BIT POSITION)
0021 10577 ;               (BITS <8> = BIT STATE)
0021 10578 ;
0021 10579 ;--
0021 10580 ;*****************************************************************************
0021 10581
0021 10582
0021 10583         INITIALIZE                                  ;INIT THE CPU
0023 10584         LDINDEX         K,1                         ;INIT INDEX K TO 1
002A 10585         LOOP            I,1,25                      ;CREATE A TEST LOOP OF 50
0031 10586         SAVEINDEX       I,1$                        ;SAVE LOOP COUNT IN 1$
0036 10587         LOADREG         WDREG,1$,,L                 ;LOAD LOOP COUNT INTO WDREG
003E 10588         ERRLOOP                                     ;ERROR LOOP POINT
0040 10589         FETCH           0                           ;START DCS PROGRAM
0045 10590         BRSTCLK         <^X0A>,INH                  ;END DCS PROGRAM
0049 10591         CMPVBUS         2$,I                        ;TEST DSIZE
004E 10592         IFERROR                                     ;ROM FAILURE
0053 10593         LOOP            J,1,5                       ;CREATE TEST LOOP OF 5
005A 10594         FETCH           <^X0B>                      ;START DCS PROGRAM
005F 10595         BRSTCLK         <^X0E>,INH                  ;END DCS PROGRAM
0063 10596         CMPVBUS         3$,K                        ;TEST DSIZE
0068 10597         IFERROR                                     ;ROM FAILURE
006D 10598         INCINDEX        K                           ;INCREMENT INDEX K BY ONE
0070 10599         ENDLOOP         J                           ;END TEST LOOP J
0073 10600         ENDLOOP         I                           ;END TEST LOOP I
```

```
                        0076 10601          SKIP                                ;GO TO NEXT TEST
                        007D 10602
                        007D 10603
                        007D 10604 BEGIN_TEST_DATA
                        007D
                        007D        ;-------------------------------------------------------------------------
                        007D 10605
                00000000 007D 10606 1$:     .LONG   ^X00000000                  ;LOOP COUNT STORAGE
                      02 0081 10607 2$:     .BYTE   ^X2                         ;RESULTS TABLE (IRD 0)
                         0082 10608         VBUSDATA        <^X24>,0            ;DECB
                         0083 10609         VBUSDATA        <^X25>,0
                      02 0084 10610         .BYTE   ^X2                         ;CVTBL
                         0085 10611         VBUSDATA        <^X24>,0
                         0086 10612         VBUSDATA        <^X25>,0
                      02 0087 10613         .BYTE   ^X2                         ;CVTBW
                         0088 10614         VBUSDATA        <^X24>,0
                         0089 10615         VBUSDATA        <^X25>,0
                      02 008A 10616         .BYTE   ^X2                         ;MOVZBL
                         008B 10617         VBUSDATA        <^X24>,0
                         008C 10618         VBUSDATA        <^X25>,0
                      02 008D 10619         .BYTE   ^X2                         ;MOVZBW
                         008E 10620         VBUSDATA        <^X24>,0
                         008F 10621         VBUSDATA        <^X25>,0
                      02 0090 10622         .BYTE   ^X2                         ;ROTL
                         0091 10623         VBUSDATA        <^X24>,0
                         0092 10624         VBUSDATA        <^X25>,0
                      02 0093 10625         .BYTE   ^X2                         ;ACBB
                         0094 10626         VBUSDATA        <^X24>,0
                         0095 10627         VBUSDATA        <^X25>,0
                      02 0096 10628         .BYTE   ^X2                         ;MOVAB
                         0097 10629         VBUSDATA        <^X24>,0
                         0098 10630         VBUSDATA        <^X25>,0
                      02 0099 10631         .BYTE   ^X2                         ;PUSHAB
                         009A 10632         VBUSDATA        <^X24>,0
                         009B 10633         VBUSDATA        <^X25>,0
                      02 009C 10634         .BYTE   ^X2                         ;ADDW2
                         009D 10635         VBUSDATA        <^X24>,1
                         009E 10636         VBUSDATA        <^X25>,0
                      02 009F 10637         .BYTE   ^X2                         ;ADDW3
                         00A0 10638         VBUSDATA        <^X24>,1
                         00A1 10639         VBUSDATA        <^X25>,0
                      02 00A2 10640         .BYTE   ^X2                         ;SUBW2
                         00A3 10641         VBUSDATA        <^X24>,1
                         00A4 10642         VBUSDATA        <^X25>,0
                      02 00A5 10643         .BYTE   ^X2                         ;SUBW3
                         00A6 10644         VBUSDATA        <^X24>,1
                         00A7 10645         VBUSDATA        <^X25>,0
                      02 00A8 10646         .BYTE   ^X2                         ;MULW2
                         00A9 10647         VBUSDATA        <^X24>,1
                         00AA 10648         VBUSDATA        <^X25>,0
                      02 00AB 10649         .BYTE   ^X2                         ;MULW3
                         00AC 10650         VBUSDATA        <^X24>,1
                         00AD 10651         VBUSDATA        <^X25>,0
                      02 00AE 10652         .BYTE   ^X2                         ;DIVW2
                         00AF 10653         VBUSDATA        <^X24>,1
```

```
                00B0 10654          VBUSDATA        <^X25>,0
02              00B1 10655          .BYTE   ^X2                             ;DIVW3
                00B2 10656          VBUSDATA        <^X24>,1
                00B3 10657          VBUSDATA        <^X25>,0
02              00B4 10658          .BYTE   ^X2                             ;BISW2
                00B5 10659          VBUSDATA        <^X24>,1
                00B6 10660          VBUSDATA        <^X25>,0
02              00B7 10661          .BYTE   ^X2                             ;BISW3
                00B8 10662          VBUSDATA        <^X24>,1
                00B9 10663          VBUSDATA        <^X25>,0
02              00BA 10664          .BYTE   ^X2                             ;BICW2
                00BB 10665          VBUSDATA        <^X24>,1
                00BC 10666          VBUSDATA        <^X25>,0
02              00BD 10667          .BYTE   ^X2                             ;BICW3
                00BE 10668          VBUSDATA        <^X24>,1
                00BF 10669          VBUSDATA        <^X25>,0
02              00C0 10670          .BYTE   ^X2                             ;XORW2
                00C1 10671          VBUSDATA        <^X24>,1
                00C2 10672          VBUSDATA        <^X25>,0
02              00C3 10673          .BYTE   ^X2                             ;XORW3
                00C4 10674          VBUSDATA        <^X24>,1
                00C5 10675          VBUSDATA        <^X25>,0
02              00C6 10676          .BYTE   ^X2                             ;MNEGW
                00C7 10677          VBUSDATA        <^X24>,1
                00C8 10678          VBUSDATA        <^X25>,0
02              00C9 10679          .BYTE   ^X2                             ;CASEW
                00CA 10680          VBUSDATA        <^X24>,1
                00CB 10681          VBUSDATA        <^X25>,0
                00CC 10682
02              00CC 10683 3$:      .BYTE   ^X2                             ;RESULTS TABLE (IRD 1 TO 5)
                00CD 10684          VBUSDATA        <^X24>,0                ;DECB
                00CE 10685          VBUSDATA        <^X25>,0
02              00CF 10686          .BYTE   ^X2
                00D0 10687          VBUSDATA        <^X24>,0
                00D1 10688          VBUSDATA        <^X25>,0
02              00D2 10689          .BYTE   ^X2
                00D3 10690          VBUSDATA        <^X24>,0
                00D4 10691          VBUSDATA        <^X25>,0
02              00D5 10692          .BYTE   ^X2
                00D6 10693          VBUSDATA        <^X24>,0
                00D7 10694          VBUSDATA        <^X25>,0
02              00D8 10695          .BYTE   ^X2
                00D9 10696          VBUSDATA        <^X24>,0
                00DA 10697          VBUSDATA        <^X25>,0
02              00DB 10698          .BYTE   ^X2                             ;CVTBL
                00DC 10699          VBUSDATA        <^X24>,0
                00DD 10700          VBUSDATA        <^X25>,1
02              00DE 10701          .BYTE   ^X2
                00DF 10702          VBUSDATA        <^X24>,0
                00E0 10703          VBUSDATA        <^X25>,0
02              00E1 10704          .BYTE   ^X2
                00E2 10705          VBUSDATA        <^X24>,0
                00E3 10706          VBUSDATA        <^X25>,0
02              00E4 10707          .BYTE   ^X2
                00E5 10708          VBUSDATA        <^X24>,0
```

```
                    00E6 10709          VBUSDATA        <^X25>,0
                02  00E7 10710          .BYTE   ^X2
                    00E8 10711          VBUSDATA        <^X24>,0
                    00E9 10712          VBUSDATA        <^X25>,0
                02  00EA 10713          .BYTE   ^X2                             ;CVTBW
                    00EB 10714          VBUSDATA        <^X24>,1
                    00EC 10715          VBUSDATA        <^X25>,0
                02  00ED 10716          .BYTE   ^X2
                    00EE 10717          VBUSDATA        <^X24>,0
                    00EF 10718          VBUSDATA        <^X25>,0
                02  00F0 10719          .BYTE   ^X2
                    00F1 10720          VBUSDATA        <^X24>,0
                    00F2 10721          VBUSDATA        <^X25>,0
                02  00F3 10722          .BYTE   ^X2
                    00F4 10723          VBUSDATA        <^X24>,0
                    00F5 10724          VBUSDATA        <^X25>,0
                02  00F6 10725          .BYTE   ^X2
                    00F7 10726          VBUSDATA        <^X24>,0
                    00F8 10727          VBUSDATA        <^X25>,0
                02  00F9 10728          .BYTE   ^X2                             ;MOVZBL
                    00FA 10729          VBUSDATA        <^X24>,0
                    00FB 10730          VBUSDATA        <^X25>,1
                02  00FC 10731          .BYTE   ^X2
                    00FD 10732          VBUSDATA        <^X24>,0
                    00FE 10733          VBUSDATA        <^X25>,0
                02  00FF 10734          .BYTE   ^X2
                    0100 10735          VBUSDATA        <^X24>,0
                    0101 10736          VBUSDATA        <^X25>,0
                02  0102 10737          .BYTE   ^X2
                    0103 10738          VBUSDATA        <^X24>,0
                    0104 10739          VBUSDATA        <^X25>,0
                02  0105 10740          .BYTE   ^X2
                    0106 10741          VBUSDATA        <^X24>,0
                    0107 10742          VBUSDATA        <^X25>,0
                02  0108 10743          .BYTE   ^X2                             ;MOVZBW
                    0109 10744          VBUSDATA        <^X24>,1
                    010A 10745          VBUSDATA        <^X25>,0
                02  010B 10746          .BYTE   ^X2
                    010C 10747          VBUSDATA        <^X24>,0
                    010D 10748          VBUSDATA        <^X25>,0
                02  010E 10749          .BYTE   ^X2
                    010F 10750          VBUSDATA        <^X24>,0
                    0110 10751          VBUSDATA        <^X25>,0
                02  0111 10752          .BYTE   ^X2
                    0112 10753          VBUSDATA        <^X24>,0
                    0113 10754          VBUSDATA        <^X25>,0
                02  0114 10755          .BYTE   ^X2
                    0115 10756          VBUSDATA        <^X24>,0
                    0116 10757          VBUSDATA        <^X25>,0
                02  0117 10758          .BYTE   ^X2                             ;ROTL
                    0118 10759          VBUSDATA        <^X24>,0
                    0119 10760          VBUSDATA        <^X25>,1
                02  011A 10761          .BYTE   ^X2
                    011B 10762          VBUSDATA        <^X24>,0
                    011C 10763          VBUSDATA        <^X25>,1
```

```
                        02  011D 10764          .BYTE   ^X2
                            011E 10765          VBUSDATA        <^X24>,0
                            011F 10766          VBUSDATA        <^X25>,0
                        02  0120 10767          .BYTE   ^X2
                            0121 10768          VBUSDATA        <^X24>,0
                            0122 10769          VBUSDATA        <^X25>,0
                        02  0123 10770          .BYTE   ^X2
                            0124 10771          VBUSDATA        <^X24>,0
                            0125 10772          VBUSDATA        <^X25>,0
                        02  0126 10773          .BYTE   ^X2                             ;ACBB
                            0127 10774          VBUSDATA        <^X24>,0
                            0128 10775          VBUSDATA        <^X25>,0
                        02  0129 10776          .BYTE   ^X2
                            012A 10777          VBUSDATA        <^X24>,0
                            012B 10778          VBUSDATA        <^X25>,0
                        02  012C 10779          .BYTE   ^X2
                            012D 10780          VBUSDATA        <^X24>,0
                            012E 10781          VBUSDATA        <^X25>,0
                        02  012F 10782          .BYTE   ^X2
                            0130 10783          VBUSDATA        <^X24>,0
                            0131 10784          VBUSDATA        <^X25>,0
                        02  0132 10785          .BYTE   ^X2
                            0133 10786          VBUSDATA        <^X24>,0
                            0134 10787          VBUSDATA        <^X25>,0
                        02  0135 10788          .BYTE   ^X2                             ;MOVAB
                            0136 10789          VBUSDATA        <^X24>,0
                            0137 10790          VBUSDATA        <^X25>,1
                        02  0138 10791          .BYTE   ^X2
                            0139 10792          VBUSDATA        <^X24>,0
                            013A 10793          VBUSDATA        <^X25>,0
                        02  013B 10794          .BYTE   ^X2
                            013C 10795          VBUSDATA        <^X24>,0
                            013D 10796          VBUSDATA        <^X25>,0
                        02  013E 10797          .BYTE   ^X2
                            013F 10798          VBUSDATA        <^X24>,0
                            0140 10799          VBUSDATA        <^X25>,0
                        02  0141 10800          .BYTE   ^X2
                            0142 10801          VBUSDATA        <^X24>,0
                            0143 10802          VBUSDATA        <^X25>,0
                        02  0144 10803          .BYTE   ^X2                             ;PUSHAB
                            0145 10804          VBUSDATA        <^X24>,0
                            0146 10805          VBUSDATA        <^X25>,0
                        02  0147 10806          .BYTE   ^X2
                            0148 10807          VBUSDATA        <^X24>,0
                            0149 10808          VBUSDATA        <^X25>,0
                        02  014A 10809          .BYTE   ^X2
                            014B 10810          VBUSDATA        <^X24>,0
                            014C 10811          VBUSDATA        <^X25>,0
                        02  014D 10812          .BYTE   ^X2
                            014E 10813          VBUSDATA        <^X24>,0
                            014F 10814          VBUSDATA        <^X25>,0
                        02  0150 10815          .BYTE   ^X2
                            0151 10816          VBUSDATA        <^X24>,0
                            0152 10817          VBUSDATA        <^X25>,0
                        02  0153 10818          .BYTE   ^X2                             ;ADDW2
```

```
                0154 10819          VBUSDATA        <^X24>,1
                0155 10820          VBUSDATA        <^X25>,0
            02  0156 10821          .BYTE   ^X2
                0157 10822          VBUSDATA        <^X24>,0
                0158 10823          VBUSDATA        <^X25>,0
            02  0159 10824          .BYTE   ^X2
                015A 10825          VBUSDATA        <^X24>,0
                015B 10826          VBUSDATA        <^X25>,0
            02  015C 10827          .BYTE   ^X2
                015D 10828          VBUSDATA        <^X24>,0
                015E 10829          VBUSDATA        <^X25>,0
            02  015F 10830          .BYTE   ^X2
                0160 10831          VBUSDATA        <^X24>,0
                0161 10832          VBUSDATA        <^X25>,0
            02  0162 10833          .BYTE   ^X2                             ;ADDW3
                0163 10834          VBUSDATA        <^X24>,1
                0164 10835          VBUSDATA        <^X25>,0
            02  0165 10836          .BYTE   ^X2
                0166 10837          VBUSDATA        <^X24>,1
                0167 10838          VBUSDATA        <^X25>,0
            02  0168 10839          .BYTE   ^X2
                0169 10840          VBUSDATA        <^X24>,0
                016A 10841          VBUSDATA        <^X25>,0
            02  016B 10842          .BYTE   ^X2
                016C 10843          VBUSDATA        <^X24>,0
                016D 10844          VBUSDATA        <^X25>,0
            02  016E 10845          .BYTE   ^X2
                016F 10846          VBUSDATA        <^X24>,0
                0170 10847          VBUSDATA        <^X25>,0
            02  0171 10848          .BYTE   ^X2                             ;SUBW2
                0172 10849          VBUSDATA        <^X24>,1
                0173 10850          VBUSDATA        <^X25>,0
            02  0174 10851          .BYTE   ^X2
                0175 10852          VBUSDATA        <^X24>,0
                0176 10853          VBUSDATA        <^X25>,0
            02  0177 10854          .BYTE   ^X2
                0178 10855          VBUSDATA        <^X24>,0
                0179 10856          VBUSDATA        <^X25>,0
            02  017A 10857          .BYTE   ^X2
                017B 10858          VBUSDATA        <^X24>,0
                017C 10859          VBUSDATA        <^X25>,0
            02  017D 10860          .BYTE   ^X2
                017E 10861          VBUSDATA        <^X24>,0
                017F 10862          VBUSDATA        <^X25>,0
            02  0180 10863          .BYTE   ^X2                             ;SUBW3
                0181 10864          VBUSDATA        <^X24>,1
                0182 10865          VBUSDATA        <^X25>,0
            02  0183 10866          .BYTE   ^X2
                0184 10867          VBUSDATA        <^X24>,1
                0185 10868          VBUSDATA        <^X25>,0
            02  0186 10869          .BYTE   ^X2
                0187 10870          VBUSDATA        <^X24>,0
                0188 10871          VBUSDATA        <^X25>,0
            02  0189 10872          .BYTE   ^X2
                018A 10873          VBUSDATA        <^X24>,0
```

```
                    018B 10874          VBUSDATA        <^X25>,0
                02  018C 10875          .BYTE   ^X2
                    018D 10876          VBUSDATA        <^X24>,0
                    018E 10877          VBUSDATA        <^X25>,0
                02  018F 10878          .BYTE   ^X2                             ;MULW2
                    0190 10879          VBUSDATA        <^X24>,1
                    0191 10880          VBUSDATA        <^X25>,0
                02  0192 10881          .BYTE   ^X2
                    0193 10882          VBUSDATA        <^X24>,0
                    0194 10883          VBUSDATA        <^X25>,0
                02  0195 10884          .BYTE   ^X2
                    0196 10885          VBUSDATA        <^X24>,0
                    0197 10886          VBUSDATA        <^X25>,0
                02  0198 10887          .BYTE   ^X2
                    0199 10888          VBUSDATA        <^X24>,0
                    019A 10889          VBUSDATA        <^X25>,0
                02  019B 10890          .BYTE   ^X2
                    019C 10891          VBUSDATA        <^X24>,0
                    019D 10892          VBUSDATA        <^X25>,0
                02  019E 10893          .BYTE   ^X2                             ;MULW3
                    019F 10894          VBUSDATA        <^X24>,1
                    01A0 10895          VBUSDATA        <^X25>,0
                02  01A1 10896          .BYTE   ^X2
                    01A2 10897          VBUSDATA        <^X24>,1
                    01A3 10898          VBUSDATA        <^X25>,0
                02  01A4 10899          .BYTE   ^X2
                    01A5 10900          VBUSDATA        <^X24>,0
                    01A6 10901          VBUSDATA        <^X25>,0
                02  01A7 10902          .BYTE   ^X2
                    01A8 10903          VBUSDATA        <^X24>,0
                    01A9 10904          VBUSDATA        <^X25>,0
                02  01AA 10905          .BYTE   ^X2
                    01AB 10906          VBUSDATA        <^X24>,0
                    01AC 10907          VBUSDATA        <^X25>,0
                02  01AD 10908          .BYTE   ^X2                             ;DIVW2
                    01AE 10909          VBUSDATA        <^X24>,1
                    01AF 10910          VBUSDATA        <^X25>,0
                02  01B0 10911          .BYTE   ^X2
                    01B1 10912          VBUSDATA        <^X24>,0
                    01B2 10913          VBUSDATA        <^X25>,0
                02  01B3 10914          .BYTE   ^X2
                    01B4 10915          VBUSDATA        <^X24>,0
                    01B5 10916          VBUSDATA        <^X25>,0
                02  01B6 10917          .BYTE   ^X2
                    01B7 10918          VBUSDATA        <^X24>,0
                    01B8 10919          VBUSDATA        <^X25>,0
                02  01B9 10920          .BYTE   ^X2
                    01BA 10921          VBUSDATA        <^X24>,0
                    01BB 10922          VBUSDATA        <^X25>,0
                02  01BC 10923          .BYTE   ^X2                             ;DIVW3
                    01BD 10924          VBUSDATA        <^X24>,1
                    01BE 10925          VBUSDATA        <^X25>,0
                02  01BF 10926          .BYTE   ^X2
                    01C0 10927          VBUSDATA        <^X24>,1
                    01C1 10928          VBUSDATA        <^X25>,0
```

```
                02  01C2 10929          .BYTE   ^X2
                    01C3 10930          VBUSDATA        <^X24>,0
                    01C4 10931          VBUSDATA        <^X25>,0
                02  01C5 10932          .BYTE   ^X2
                    01C6 10933          VBUSDATA        <^X24>,0
                    01C7 10934          VBUSDATA        <^X25>,0
                02  01C8 10935          .BYTE   ^X2
                    01C9 10936          VBUSDATA        <^X24>,0
                    01CA 10937          VBUSDATA        <^X25>,0
                02  01CB 10938          .BYTE   ^X2                             ;BISW2
                    01CC 10939          VBUSDATA        <^X24>,1
                    01CD 10940          VBUSDATA        <^X25>,0
                02  01CE 10941          .BYTE   ^X2
                    01CF 10942          VBUSDATA        <^X24>,0
                    01D0 10943          VBUSDATA        <^X25>,0
                02  01D1 10944          .BYTE   ^X2
                    01D2 10945          VBUSDATA        <^X24>,0
                    01D3 10946          VBUSDATA        <^X25>,0
                02  01D4 10947          .BYTE   ^X2
                    01D5 10948          VBUSDATA        <^X24>,0
                    01D6 10949          VBUSDATA        <^X25>,0
                02  01D7 10950          .BYTE   ^X2
                    01D8 10951          VBUSDATA        <^X24>,0
                    01D9 10952          VBUSDATA        <^X25>,0
                02  01DA 10953          .BYTE   ^X2                             ;BISW3
                    01DB 10954          VBUSDATA        <^X24>,1
                    01DC 10955          VBUSDATA        <^X25>,0
                02  01DD 10956          .BYTE   ^X2
                    01DE 10957          VBUSDATA        <^X24>,1
                    01DF 10958          VBUSDATA        <^X25>,0
                02  01E0 10959          .BYTE   ^X2
                    01E1 10960          VBUSDATA        <^X24>,0
                    01E2 10961          VBUSDATA        <^X25>,0
                02  01E3 10962          .BYTE   ^X2
                    01E4 10963          VBUSDATA        <^X24>,0
                    01E5 10964          VBUSDATA        <^X25>,0
                02  01E6 10965          .BYTE   ^X2
                    01E7 10966          VBUSDATA        <^X24>,0
                    01E8 10967          VBUSDATA        <^X25>,0
                02  01E9 10968          .BYTE   ^X2                             ;BICW2
                    01EA 10969          VBUSDATA        <^X24>,1
                    01EB 10970          VBUSDATA        <^X25>,0
                02  01EC 10971          .BYTE   ^X2
                    01ED 10972          VBUSDATA        <^X24>,0
                    01EE 10973          VBUSDATA        <^X25>,0
                02  01EF 10974          .BYTE   ^X2
                    01F0 10975          VBUSDATA        <^X24>,0
                    01F1 10976          VBUSDATA        <^X25>,0
                02  01F2 10977          .BYTE   ^X2
                    01F3 10978          VBUSDATA        <^X24>,0
                    01F4 10979          VBUSDATA        <^X25>,0
                02  01F5 10980          .BYTE   ^X2
                    01F6 10981          VBUSDATA        <^X24>,0
                    01F7 10982          VBUSDATA        <^X25>,0
                02  01F8 10983          .BYTE   ^X2                             ;BICW3
```

```
                    01F9 10984          VBUSDATA        <^X24>,1
                    01FA 10985          VBUSDATA        <^X25>,0
                02  01FB 10986          .BYTE   ^X2
                    01FC 10987          VBUSDATA        <^X24>,1
                    01FD 10988          VBUSDATA        <^X25>,0
                02  01FE 10989          .BYTE   ^X2
                    01FF 10990          VBUSDATA        <^X24>,0
                    0200 10991          VBUSDATA        <^X25>,0
                02  0201 10992          .BYTE   ^X2
                    0202 10993          VBUSDATA        <^X24>,0
                    0203 10994          VBUSDATA        <^X25>,0
                02  0204 10995          .BYTE   ^X2
                    0205 10996          VBUSDATA        <^X24>,0
                    0206 10997          VBUSDATA        <^X25>,0
                02  0207 10998          .BYTE   ^X2                             ;XORW2
                    0208 10999          VBUSDATA        <^X24>,1
                    0209 11000          VBUSDATA        <^X25>,0
                02  020A 11001          .BYTE   ^X2
                    020B 11002          VBUSDATA        <^X24>,0
                    020C 11003          VBUSDATA        <^X25>,0
                02  020D 11004          .BYTE   ^X2
                    020E 11005          VBUSDATA        <^X24>,0
                    020F 11006          VBUSDATA        <^X25>,0
                02  0210 11007          .BYTE   ^X2
                    0211 11008          VBUSDATA        <^X24>,0
                    0212 11009          VBUSDATA        <^X25>,0
                02  0213 11010          .BYTE   ^X2
                    0214 11011          VBUSDATA        <^X24>,0
                    0215 11012          VBUSDATA        <^X25>,0
                02  0216 11013          .BYTE   ^X2                             ;XORW3
                    0217 11014          VBUSDATA        <^X24>,1
                    0218 11015          VBUSDATA        <^X25>,0
                02  0219 11016          .BYTE   ^X2
                    021A 11017          VBUSDATA        <^X24>,1
                    021B 11018          VBUSDATA        <^X25>,0
                02  021C 11019          .BYTE   ^X2
                    021D 11020          VBUSDATA        <^X24>,0
                    021E 11021          VBUSDATA        <^X25>,0
                02  021F 11022          .BYTE   ^X2
                    0220 11023          VBUSDATA        <^X24>,0
                    0221 11024          VBUSDATA        <^X25>,0
                02  0222 11025          .BYTE   ^X2
                    0223 11026          VBUSDATA        <^X24>,0
                    0224 11027          VBUSDATA        <^X25>,0
                02  0225 11028          .BYTE   ^X2                             ;MNEGW
                    0226 11029          VBUSDATA        <^X24>,1
                    0227 11030          VBUSDATA        <^X25>,0
                02  0228 11031          .BYTE   ^X2
                    0229 11032          VBUSDATA        <^X24>,0
                    022A 11033          VBUSDATA        <^X25>,0
                02  022B 11034          .BYTE   ^X2
                    022C 11035          VBUSDATA        <^X24>,0
                    022D 11036          VBUSDATA        <^X25>,0
                02  022E 11037          .BYTE   ^X2
                    022F 11038          VBUSDATA        <^X24>,0
```

```
                                0230 11039          VBUSDATA        <^X25>,0
                             02 0231 11040          .BYTE   ^X2
                                0232 11041          VBUSDATA        <^X24>,0
                                0233 11042          VBUSDATA        <^X25>,0
                             02 0234 11043          .BYTE   ^X2                             ;CASEW
                                0235 11044          VBUSDATA        <^X24>,1
                                0236 11045          VBUSDATA        <^X25>,0
                             02 0237 11046          .BYTE   ^X2
                                0238 11047          VBUSDATA        <^X24>,1
                                0239 11048          VBUSDATA        <^X25>,0
                             02 023A 11049          .BYTE   ^X2
                                023B 11050          VBUSDATA        <^X24>,0
                                023C 11051          VBUSDATA        <^X25>,0
                             02 023D 11052          .BYTE   ^X2
                                023E 11053          VBUSDATA        <^X24>,0
                                023F 11054          VBUSDATA        <^X25>,0
                             02 0240 11055          .BYTE   ^X2
                                0241 11056          VBUSDATA        <^X24>,0
                                0242 11057          VBUSDATA        <^X25>,0
                                0243 11058
                                0243 11059
                                0243 11060
                                0243 11061 BEGIN_CPU_DATA
                                0002 11062
                                0002 11063 DCS_DATA
                                0001 11064
U 00, 7700,C800,0364,0300,0470,1801  0001 11065 ;% 00   NOP
U 01, 7700,4800,5BE4,0300,2470,1802  000C 11069 ;% 01   WCTRL/LOADCRAR,WB_R[TEMP0]          ;DISABLE ^P INTERRUPT
U 02, 7700,0800,5BE4,0304,2C70,1803  0017 11073 ;% 02   WCTRL/CONWRITE,WB_R[TEMP1]         ;...
U 03, 7700,9800,D370,030F,FA70,1804  0022 11077 ;% 03   IPL_[1F]                           ;SET IPL TO 1F
U 04, 5700,4840,0364,0308,0470,1805  002D 11081 ;% 04   R[TEMP2]_RDM                       ;MOVE LOOP COUNT TO RTEMP2
U 05, 7700,D800,EF64,0301,0470,1806  0038 11085 ;% 05   PL_[2]                            ;SET PLATCH TO 2
U 06, 7700,C800,8B71,0308,0470,1807  0043 11089 ;% 06   Q_R[TEMP2].RL.P                   ;ROTATE LOOP COUNT LEFT 2
U 07, 7700,4800,03A4,03D8,4870,1808  004E 11093 ;% 07   WCTRL/PC_WB,WB_Q                   ;FORCE PREFETCH TO HAPPEN
U 08, 7700,4800,0364,1700,0470,1809  0059 11097 ;% 08   IRD1TEST                          ;LOAD IR REGISTER
U 09, 7600,4800,0364,0300,0470,180A  0064 11101 ;% 09   DTYPE/IDEP,STROBE.V.BUS           ;LOAD DSIZE INTO VBUS
U 0A, 7300,C800,0364,0300,0470,180B  006F 11105 ;% 0A   NOP,STOP.CLOCK                   ;STOP THE CPU CLOCK
U 0B, 7700,4800,0364,0300,0470,180C  007A 11109 ;% 0B   NOP
U 0C, 7700,C800,0364,1B00,0470,180D  0085 11113 ;% 0C   LOD INC BRA?                     ;INCREMENT THE IRD COUNTER
U 0D, 7600,C800,0364,0300,0470,180E  0090 11117 ;% 0D   DTYPE/IDEP,STROBE.V.BUS           ;LOAD DSIZE INTO VBUS
U 0E, 7300,4800,0364,0300,0470,180F  009B 11121 ;% 0E   NOP,STOP.CLOCK                   ;STOP THE CPU CLOCK
                                00A6 11125
                                00A6 11126 RTEMP_DATA
                                0001 11127
                      00C00000  0001 11128          .LONG   ^X00C00000                    ;DISABLE ^P INTERRUPT
                      00400000  0005 11129          .LONG   ^X00400000                    ;...
                                0009 11130
                                0009 11131 CACHE_DATA
                                0001 11132
                      00000097  0001 11133          .LONG   ^X00000097                    ;DECB
                      00000098  0005 11134          .LONG   ^X00000098                    ;CVTBL
                      00000099  0009 11135          .LONG   ^X00000099                    ;CVTBW
                      0000009A  000D 11136          .LONG   ^X0000009A                    ;MOVZBL
                      0000009B  0011 11137          .LONG   ^X0000009B                    ;MOVZBW
                      0000009C  0015 11138          .LONG   ^X0000009C                    ;ROTL
```

```
0000009D  0019 11139          .LONG   ^X0000009D                      ;ACBB
0000009E  001D 11140          .LONG   ^X0000009E                      ;MOVAB
0000009F  0021 11141          .LONG   ^X0000009F                      ;PUSHAB
000000A0  0025 11142          .LONG   ^X000000A0                      ;ADDW2
000000A1  0029 11143          .LONG   ^X000000A1                      ;ADDW3
000000A2  002D 11144          .LONG   ^X000000A2                      ;SUBW2
000000A3  0031 11145          .LONG   ^X000000A3                      ;SUBW3
000000A4  0035 11146          .LONG   ^X000000A4                      ;MULW2
000000A5  0039 11147          .LONG   ^X000000A5                      ;MULW3
000000A6  003D 11148          .LONG   ^X000000A6                      ;DIVW2
000000A7  0041 11149          .LONG   ^X000000A7                      ;DIVW3
000000A8  0045 11150          .LONG   ^X000000A8                      ;BISW2
000000A9  0049 11151          .LONG   ^X000000A9                      ;BISW3
000000AA  004D 11152          .LONG   ^X000000AA                      ;BICW2
000000AB  0051 11153          .LONG   ^X000000AB                      ;BICW3
000000AC  0055 11154          .LONG   ^X000000AC                      ;XORW2
000000AD  0059 11155          .LONG   ^X000000AD                      ;XORW3
000000AE  005D 11156          .LONG   ^X000000AE                      ;MNEGW
000000AF  0061 11157          .LONG   ^X000000AF                      ;CASEW
00000000  0065 11158          .LONG   ^X00000000
          0069 11159
          0069 11160 END_CPU_DATA
          0243 11161
          0243 11162
          0243 11163 END_TEST_DATA
          0243
          0243          ;-----------------------------------------------------------------------------
          0243          ;-----------------------------------------------------------------------------
          0243 11164 ENDTEST
```

```
                        0021        .SBTTL  "TEST   08B     NATIVE MODE DSIZE ROM TEST 9
                        0021 11166 ;*******************************************************************************
                        0021 11167 ;++
                        0021 11168 ;
                        0021 11169 ; FUNCTIONAL DESCRIPTION:
                        0021 11170 ;
                        0021 11171 ;       THIS IS ONE IN A SERIES OF 14 TESTS THAT CHECK THE CONTENTS OF THE
                        0021 11172 ;       DSIZE ROM.
                        0021 11173 ;
                        0021 11174 ;
                        0021 11175 ; TEST ALGORITHM:
                        0021 11176 ;
                        0021 11177 ;       1.  CREATE AN INDEX (K) TO INDEX RESULTS TABLE
                        0021 11178 ;       2.  CREATE LOOP (I) OF 25
                        0021 11179 ;       3.  SAVE LOOP COUNT IN 1$
                        0021 11180 ;       4.  MOVE LOOP COUNT TO WDREG
                        0021 11181 ;       5.  EXECUTE DCS PROGRAM
                        0021 11182 ;               a.  DISABLE ^P INTERRUPTS
                        0021 11183 ;               b.  SET IPL TO 1F
                        0021 11184 ;               c.  MOVE LOOP COUNT TO RTEMP2
                        0021 11185 ;               d.  ROTATE LOOP COUNT LEFT 2 TO FORM CACHE ADDRESS
                        0021 11186 ;               e.  LOAD THIS ADDRESS INTO PC TO CAUSE PREFETCH
                        0021 11187 ;               f.  PERFORM IRD1TEST TO CAUSE IR TO LOAD INSTRUCTION
                        0021 11188 ;               g.  SET DTYPE TO INSTURCTION DEPENDENT AND STROBE THE VBUS
                        0021 11189 ;       6.  TEST VBUS FOR CORRECT DSIZE
                        0021 11190 ;       7.  IF NO ERROR CREATE LOOP (J) OF 5
                        0021 11191 ;       8.  EXECUTE DCS PROGRAM
                        0021 11192 ;               a.  INCREMENT THE IRD COUNTER
                        0021 11193 ;               b.  SET DTYPE TO INSTRUCTION DEPENDENT AND STROBE THE VBUS
                        0021 11194 ;       9.  TEST VBUS FOR CORRECT DSIZE
                        0021 11195 ;      10.  INCREMENT INDEX K BY ONE
                        0021 11196 ;      11.  IF NO ERROR GO TO STEP 8 UNTIL LOOP J IS EXHAUSTED
                        0021 11197 ;      12.  GO TO STEP 3 UNTIL LOOP I IS EXHAUSTED
                        0021 11198 ;
                        0021 11199 ;
                        0021 11200 ; TEST PATTERNS:
                        0021 11201 ;
                        0021 11202 ;       LOOP COUNT I             INSTRUCTION
                        0021 11203 ;            1                      MOVW
                        0021 11204 ;            2                      CMPW
                        0021 11205 ;            3                      MCOMW
                        0021 11206 ;            4                      BITW
                        0021 11207 ;            5                      CLRW
                        0021 11208 ;            6                      TSTW
                        0021 11209 ;            7                      INCW
                        0021 11210 ;            8                      DECW
                        0021 11211 ;            9                      BISPSW
                        0021 11212 ;            A                      BICPSW
                        0021 11213 ;            B                      POPR
                        0021 11214 ;            C                      PUSHR
                        0021 11215 ;            D                      CHMK
                        0021 11216 ;            E                      CHME
                        0021 11217 ;            F                      CHMS
                        0021 11218 ;           10                      CHMU
                        0021 11219 ;           11                      ADDL2
```

```
0021 11220 ;           12                   ADDL3
0021 11221 ;           13                   SUBL2
0021 11222 ;           14                   SUBL3
0021 11223 ;           15                   MULL2
0021 11224 ;           16                   MULL3
0021 11225 ;           17                   DIVL2
0021 11226 ;           18                   DIVL3
0021 11227 ;           19                   BISL2
0021 11228 ;
0021 11229 ;
0021 11230 ; LOGIC DESCRIPTION:
0021 11231 ;
0021 11232 ;       THIS TEST ALONG WITH THE OTHER 13 TESTS IN THIS SERIES CHECK ALL THE
0021 11233 ;       LOCATIONS BLASTED INTO THE DSIZE ROM.  ANY FALIURES SHOULD BE CONFINED
0021 11234 ;       TO THE ROM.
0021 11235 ;
0021 11236 ;
0021 11237 ; ASSUMPTIONS:
0021 11238 ;
0021 11239 ;       THIS TEST ASSUMES THE PREFETCH LOGIC AND INSTRUCTION REGISTER ARE
0021 11240 ;       OPERATIONAL.
0021 11241 ;
0021 11242 ;
0021 11243 ; ERROR DESCRIPTION:
0021 11244 ;
0021 11245 ;       EXPECTED VBUS DATA (DSIZE)
0021 11246 ;       RECEIVED VBUS DATA
0021 11247 ;       LOOP COUNT I
0021 11248 ;       LOOP COUNT J
0021 11249 ;       LOOP COUNT K
0021 11250 ;               (BITS <7:0> = VBUS BIT POSITION)
0021 11251 ;               (BITS <8> = BIT STATE)
0021 11252 ;
0021 11253 ;--
0021 11254 ;*******************************************************************************
0021 11255
0021 11256
0021 11257          INITIALIZE                              ;INIT THE CPU
0023 11258          LDINDEX         K,1                     ;INIT INDEX K TO 1
002A 11259          LOOP            I,1,25                  ;CREATE A TEST LOOP OF 50
0031 11260          SAVEINDEX       I,1$                    ;SAVE LOOP COUNT IN 1$
0036 11261          LOADREG         WDREG,1$,,L             ;LOAD LOOP COUNT INTO WDREG
003E 11262          ERRLOOP                                 ;ERROR LOOP POINT
0040 11263          FETCH           0                       ;START DCS PROGRAM
0045 11264          BRSTCLK         <^X0A>,INH              ;END DCS PROGRAM
0049 11265          CMPVBUS         2$,I                    ;TEST DSIZE
004E 11266          IFERROR                                 ;ROM FAILURE
0053 11267          LOOP            J,1,5                   ;CREATE TEST LOOP OF 5
005A 11268          FETCH           <^X0B>                  ;START DCS PROGRAM
005F 11269          BRSTCLK         <^X0E>,INH              ;END DCS PROGRAM
0063 11270          CMPVBUS         3$,K                    ;TEST DSIZE
0068 11271          IFERROR                                 ;ROM FAILURE
006D 11272          INCINDEX        K                       ;INCREMENT INDEX K BY ONE
0070 11273          ENDLOOP         J                       ;END TEST LOOP J
0073 11274          ENDLOOP         I                       ;END TEST LOOP I
```

```
                     0076 11275          SKIP                                        ;GO TO NEXT TEST
                     007D 11276
                     007D 11277
                     007D 11278 BEGIN_TEST_DATA
                     007D
                     007D        ;-------------------------------------------------------------------------
                     007D 11279
                00000000  007D 11280 1$:      .LONG    ^X00000000                    ;LOOP COUNT STORAGE
                      02  0081 11281 2$:      .BYTE    ^X2                           ;RESULTS TABLE (IRD 0)
                          0082 11282          VBUSDATA         <^X24>,1              ;MOVW
                          0083 11283          VBUSDATA         <^X25>,0
                      02  0084 11284          .BYTE    ^X2                           ;CMPW
                          0085 11285          VBUSDATA         <^X24>,1
                          0086 11286          VBUSDATA         <^X25>,0
                      02  0087 11287          .BYTE    ^X2                           ;MCOMW
                          0088 11288          VBUSDATA         <^X24>,1
                          0089 11289          VBUSDATA         <^X25>,0
                      02  008A 11290          .BYTE    ^X2                           ;BITW
                          008B 11291          VBUSDATA         <^X24>,1
                          008C 11292          VBUSDATA         <^X25>,0
                      02  008D 11293          .BYTE    ^X2                           ;CLRW
                          008E 11294          VBUSDATA         <^X24>,1
                          008F 11295          VBUSDATA         <^X25>,0
                      02  0090 11296          .BYTE    ^X2                           ;TSTW
                          0091 11297          VBUSDATA         <^X24>,1
                          0092 11298          VBUSDATA         <^X25>,0
                      02  0093 11299          .BYTE    ^X2                           ;INCW
                          0094 11300          VBUSDATA         <^X24>,1
                          0095 11301          VBUSDATA         <^X25>,0
                      02  0096 11302          .BYTE    ^X2                           ;DECW
                          0097 11303          VBUSDATA         <^X24>,1
                          0098 11304          VBUSDATA         <^X25>,0
                      02  0099 11305          .BYTE    ^X2                           ;BISPSW
                          009A 11306          VBUSDATA         <^X24>,1
                          009B 11307          VBUSDATA         <^X25>,0
                      02  009C 11308          .BYTE    ^X2                           ;BICPSW
                          009D 11309          VBUSDATA         <^X24>,1
                          009E 11310          VBUSDATA         <^X25>,0
                      02  009F 11311          .BYTE    ^X2                           ;POPR
                          00A0 11312          VBUSDATA         <^X24>,1
                          00A1 11313          VBUSDATA         <^X25>,0
                      02  00A2 11314          .BYTE    ^X2                           ;PUSHR
                          00A3 11315          VBUSDATA         <^X24>,1
                          00A4 11316          VBUSDATA         <^X25>,0
                      02  00A5 11317          .BYTE    ^X2                           ;CHMK
                          00A6 11318          VBUSDATA         <^X24>,1
                          00A7 11319          VBUSDATA         <^X25>,0
                      02  00A8 11320          .BYTE    ^X2                           ;CHME
                          00A9 11321          VBUSDATA         <^X24>,1
                          00AA 11322          VBUSDATA         <^X25>,0
                      02  00AB 11323          .BYTE    ^X2                           ;CHMS
                          00AC 11324          VBUSDATA         <^X24>,1
                          00AD 11325          VBUSDATA         <^X25>,0
                      02  00AE 11326          .BYTE    ^X2                           ;CHMU
                          00AF 11327          VBUSDATA         <^X24>,1
```

```
                    00B0 11328          VBUSDATA        <^X25>,0
02                  00B1 11329          .BYTE   ^X2                             ;ADDL2
                    00B2 11330          VBUSDATA        <^X24>,0
                    00B3 11331          VBUSDATA        <^X25>,1
02                  00B4 11332          .BYTE   ^X2                             ;ADDL3
                    00B5 11333          VBUSDATA        <^X24>,0
                    00B6 11334          VBUSDATA        <^X25>,1
02                  00B7 11335          .BYTE   ^X2                             ;SUBL2
                    00B8 11336          VBUSDATA        <^X24>,0
                    00B9 11337          VBUSDATA        <^X25>,1
02                  00BA 11338          .BYTE   ^X2                             ;SUBL3
                    00BB 11339          VBUSDATA        <^X24>,0
                    00BC 11340          VBUSDATA        <^X25>,1
02                  00BD 11341          .BYTE   ^X2                             ;MULL2
                    00BE 11342          VBUSDATA        <^X24>,0
                    00BF 11343          VBUSDATA        <^X25>,1
02                  00C0 11344          .BYTE   ^X2                             ;MULL3
                    00C1 11345          VBUSDATA        <^X24>,0
                    00C2 11346          VBUSDATA        <^X25>,1
02                  00C3 11347          .BYTE   ^X2                             ;DIVL2
                    00C4 11348          VBUSDATA        <^X24>,0
                    00C5 11349          VBUSDATA        <^X25>,1
02                  00C6 11350          .BYTE   ^X2                             ;DIVL3
                    00C7 11351          VBUSDATA        <^X24>,0
                    00C8 11352          VBUSDATA        <^X25>,1
02                  00C9 11353          .BYTE   ^X2                             ;BISL2
                    00CA 11354          VBUSDATA        <^X24>,0
                    00CB 11355          VBUSDATA        <^X25>,1
                    00CC 11356
02                  00CC 11357 3$:      .BYTE   ^X2                             ;RESULTS TABLE (IRD 1 TO 5)
                    00CD 11358          VBUSDATA        <^X24>,1                ;MOVW
                    00CE 11359          VBUSDATA        <^X25>,0
02                  00CF 11360          .BYTE   ^X2
                    00D0 11361          VBUSDATA        <^X24>,0
                    00D1 11362          VBUSDATA        <^X25>,0
02                  00D2 11363          .BYTE   ^X2
                    00D3 11364          VBUSDATA        <^X24>,0
                    00D4 11365          VBUSDATA        <^X25>,0
02                  00D5 11366          .BYTE   ^X2
                    00D6 11367          VBUSDATA        <^X24>,0
                    00D7 11368          VBUSDATA        <^X25>,0
02                  00D8 11369          .BYTE   ^X2
                    00D9 11370          VBUSDATA        <^X24>,0
                    00DA 11371          VBUSDATA        <^X25>,0
02                  00DB 11372          .BYTE   ^X2                             ;CMPW
                    00DC 11373          VBUSDATA        <^X24>,1
                    00DD 11374          VBUSDATA        <^X25>,0
02                  00DE 11375          .BYTE   ^X2
                    00DF 11376          VBUSDATA        <^X24>,0
                    00E0 11377          VBUSDATA        <^X25>,0
02                  00E1 11378          .BYTE   ^X2
                    00E2 11379          VBUSDATA        <^X24>,0
                    00E3 11380          VBUSDATA        <^X25>,0
02                  00E4 11381          .BYTE   ^X2
                    00E5 11382          VBUSDATA        <^X24>,0
```

```
                    00E6 11383          VBUSDATA        <^X25>,0
                02  00E7 11384          .BYTE   ^X2
                    00E8 11385          VBUSDATA        <^X24>,0
                    00E9 11386          VBUSDATA        <^X25>,0
                02  00EA 11387          .BYTE   ^X2                             ;MCOMW
                    00EB 11388          VBUSDATA        <^X24>,1
                    00EC 11389          VBUSDATA        <^X25>,0
                02  00ED 11390          .BYTE   ^X2
                    00EE 11391          VBUSDATA        <^X24>,0
                    00EF 11392          VBUSDATA        <^X25>,0
                02  00F0 11393          .BYTE   ^X2
                    00F1 11394          VBUSDATA        <^X24>,0
                    00F2 11395          VBUSDATA        <^X25>,0
                02  00F3 11396          .BYTE   ^X2
                    00F4 11397          VBUSDATA        <^X24>,0
                    00F5 11398          VBUSDATA        <^X25>,0
                02  00F6 11399          .BYTE   ^X2
                    00F7 11400          VBUSDATA        <^X24>,0
                    00F8 11401          VBUSDATA        <^X25>,0
                02  00F9 11402          .BYTE   ^X2                             ;BITW
                    00FA 11403          VBUSDATA        <^X24>,1
                    00FB 11404          VBUSDATA        <^X25>,0
                02  00FC 11405          .BYTE   ^X2
                    00FD 11406          VBUSDATA        <^X24>,0
                    00FE 11407          VBUSDATA        <^X25>,0
                02  00FF 11408          .BYTE   ^X2
                    0100 11409          VBUSDATA        <^X24>,0
                    0101 11410          VBUSDATA        <^X25>,0
                02  0102 11411          .BYTE   ^X2
                    0103 11412          VBUSDATA        <^X24>,0
                    0104 11413          VBUSDATA        <^X25>,0
                02  0105 11414          .BYTE   ^X2
                    0106 11415          VBUSDATA        <^X24>,0
                    0107 11416          VBUSDATA        <^X25>,0
                02  0108 11417          .BYTE   ^X2                             ;CLRW
                    0109 11418          VBUSDATA        <^X24>,0
                    010A 11419          VBUSDATA        <^X25>,0
                02  010B 11420          .BYTE   ^X2
                    010C 11421          VBUSDATA        <^X24>,0
                    010D 11422          VBUSDATA        <^X25>,0
                02  010E 11423          .BYTE   ^X2
                    010F 11424          VBUSDATA        <^X24>,0
                    0110 11425          VBUSDATA        <^X25>,0
                02  0111 11426          .BYTE   ^X2
                    0112 11427          VBUSDATA        <^X24>,0
                    0113 11428          VBUSDATA        <^X25>,0
                02  0114 11429          .BYTE   ^X2
                    0115 11430          VBUSDATA        <^X24>,0
                    0116 11431          VBUSDATA        <^X25>,0
                02  0117 11432          .BYTE   ^X2                             ;TSTW
                    0118 11433          VBUSDATA        <^X24>,0
                    0119 11434          VBUSDATA        <^X25>,0
                02  011A 11435          .BYTE   ^X2
                    011B 11436          VBUSDATA        <^X24>,0
                    011C 11437          VBUSDATA        <^X25>,0
```

```
                02  011D 11438          .BYTE   ^X2
                    011E 11439          VBUSDATA        <^X24>,0
                    011F 11440          VBUSDATA        <^X25>,0
                02  0120 11441          .BYTE   ^X2
                    0121 11442          VBUSDATA        <^X24>,0
                    0122 11443          VBUSDATA        <^X25>,0
                02  0123 11444          .BYTE   ^X2
                    0124 11445          VBUSDATA        <^X24>,0
                    0125 11446          VBUSDATA        <^X25>,0
                02  0126 11447          .BYTE   ^X2                             ;INCW
                    0127 11448          VBUSDATA        <^X24>,0
                    0128 11449          VBUSDATA        <^X25>,0
                02  0129 11450          .BYTE   ^X2
                    012A 11451          VBUSDATA        <^X24>,0
                    012B 11452          VBUSDATA        <^X25>,0
                02  012C 11453          .BYTE   ^X2
                    012D 11454          VBUSDATA        <^X24>,0
                    012E 11455          VBUSDATA        <^X25>,0
                02  012F 11456          .BYTE   ^X2
                    0130 11457          VBUSDATA        <^X24>,0
                    0131 11458          VBUSDATA        <^X25>,0
                02  0132 11459          .BYTE   ^X2
                    0133 11460          VBUSDATA        <^X24>,0
                    0134 11461          VBUSDATA        <^X25>,0
                02  0135 11462          .BYTE   ^X2                             ;DECW
                    0136 11463          VBUSDATA        <^X24>,0
                    0137 11464          VBUSDATA        <^X25>,0
                02  0138 11465          .BYTE   ^X2
                    0139 11466          VBUSDATA        <^X24>,0
                    013A 11467          VBUSDATA        <^X25>,0
                02  013B 11468          .BYTE   ^X2
                    013C 11469          VBUSDATA        <^X24>,0
                    013D 11470          VBUSDATA        <^X25>,0
                02  013E 11471          .BYTE   ^X2
                    013F 11472          VBUSDATA        <^X24>,0
                    0140 11473          VBUSDATA        <^X25>,0
                02  0141 11474          .BYTE   ^X2
                    0142 11475          VBUSDATA        <^X24>,0
                    0143 11476          VBUSDATA        <^X25>,0
                02  0144 11477          .BYTE   ^X2                         ;BISPSW
                    0145 11478          VBUSDATA        <^X24>,0
                    0146 11479          VBUSDATA        <^X25>,0
                02  0147 11480          .BYTE   ^X2
                    0148 11481          VBUSDATA        <^X24>,0
                    0149 11482          VBUSDATA        <^X25>,0
                02  014A 11483          .BYTE   ^X2
                    014B 11484          VBUSDATA        <^X24>,0
                    014C 11485          VBUSDATA        <^X25>,0
                02  014D 11486          .BYTE   ^X2
                    014E 11487          VBUSDATA        <^X24>,0
                    014F 11488          VBUSDATA        <^X25>,0
                02  0150 11489          .BYTE   ^X2
                    0151 11490          VBUSDATA        <^X24>,0
                    0152 11491          VBUSDATA        <^X25>,0
                02  0153 11492          .BYTE   ^X2                         ;BICPSW
```

```
                0154 11493          VBUSDATA        <^X24>,0
                0155 11494          VBUSDATA        <^X25>,0
          02    0156 11495          .BYTE   ^X2
                0157 11496          VBUSDATA        <^X24>,0
                0158 11497          VBUSDATA        <^X25>,0
          02    0159 11498          .BYTE   ^X2
                015A 11499          VBUSDATA        <^X24>,0
                015B 11500          VBUSDATA        <^X25>,0
          02    015C 11501          .BYTE   ^X2
                015D 11502          VBUSDATA        <^X24>,0
                015E 11503          VBUSDATA        <^X25>,0
          02    015F 11504          .BYTE   ^X2
                0160 11505          VBUSDATA        <^X24>,0
                0161 11506          VBUSDATA        <^X25>,0
          02    0162 11507          .BYTE   ^X2                             ;POPR
                0163 11508          VBUSDATA        <^X24>,0
                0164 11509          VBUSDATA        <^X25>,0
          02    0165 11510          .BYTE   ^X2
                0166 11511          VBUSDATA        <^X24>,0
                0167 11512          VBUSDATA        <^X25>,0
          02    0168 11513          .BYTE   ^X2
                0169 11514          VBUSDATA        <^X24>,0
                016A 11515          VBUSDATA        <^X25>,0
          02    016B 11516          .BYTE   ^X2
                016C 11517          VBUSDATA        <^X24>,0
                016D 11518          VBUSDATA        <^X25>,0
          02    016E 11519          .BYTE   ^X2
                016F 11520          VBUSDATA        <^X24>,0
                0170 11521          VBUSDATA        <^X25>,0
          02    0171 11522          .BYTE   ^X2                         ;PUSHR
                0172 11523          VBUSDATA        <^X24>,0
                0173 11524          VBUSDATA        <^X25>,0
          02    0174 11525          .BYTE   ^X2
                0175 11526          VBUSDATA        <^X24>,0
                0176 11527          VBUSDATA        <^X25>,0
          02    0177 11528          .BYTE   ^X2
                0178 11529          VBUSDATA        <^X24>,0
                0179 11530          VBUSDATA        <^X25>,0
          02    017A 11531          .BYTE   ^X2
                017B 11532          VBUSDATA        <^X24>,0
                017C 11533          VBUSDATA        <^X25>,0
          02    017D 11534          .BYTE   ^X2
                017E 11535          VBUSDATA        <^X24>,0
                017F 11536          VBUSDATA        <^X25>,0
          02    0180 11537          .BYTE   ^X2                         ;CHMK
                0181 11538          VBUSDATA        <^X24>,0
                0182 11539          VBUSDATA        <^X25>,0
          02    0183 11540          .BYTE   ^X2
                0184 11541          VBUSDATA        <^X24>,0
                0185 11542          VBUSDATA        <^X25>,0
          02    0186 11543          .BYTE   ^X2
                0187 11544          VBUSDATA        <^X24>,0
                0188 11545          VBUSDATA        <^X25>,0
          02    0189 11546          .BYTE   ^X2
                018A 11547          VBUSDATA        <^X24>,0
```

```
                018B 11548          VBUSDATA        <^X25>,0
            02  018C 11549          .BYTE   ^X2
                018D 11550          VBUSDATA        <^X24>,0
                018E 11551          VBUSDATA        <^X25>,0
            02  018F 11552          .BYTE   ^X2                             ;CHME
                0190 11553          VBUSDATA        <^X24>,0
                0191 11554          VBUSDATA        <^X25>,0
            02  0192 11555          .BYTE   ^X2
                0193 11556          VBUSDATA        <^X24>,0
                0194 11557          VBUSDATA        <^X25>,0
            02  0195 11558          .BYTE   ^X2
                0196 11559          VBUSDATA        <^X24>,0
                0197 11560          VBUSDATA        <^X25>,0
            02  0198 11561          .BYTE   ^X2
                0199 11562          VBUSDATA        <^X24>,0
                019A 11563          VBUSDATA        <^X25>,0
            02  019B 11564          .BYTE   ^X2
                019C 11565          VBUSDATA        <^X24>,0
                019D 11566          VBUSDATA        <^X25>,0
            02  019E 11567          .BYTE   ^X2                             ;CHMS
                019F 11568          VBUSDATA        <^X24>,0
                01A0 11569          VBUSDATA        <^X25>,0
            02  01A1 11570          .BYTE   ^X2
                01A2 11571          VBUSDATA        <^X24>,0
                01A3 11572          VBUSDATA        <^X25>,0
            02  01A4 11573          .BYTE   ^X2
                01A5 11574          VBUSDATA        <^X24>,0
                01A6 11575          VBUSDATA        <^X25>,0
            02  01A7 11576          .BYTE   ^X2
                01A8 11577          VBUSDATA        <^X24>,0
                01A9 11578          VBUSDATA        <^X25>,0
            02  01AA 11579          .BYTE   ^X2
                01AB 11580          VBUSDATA        <^X24>,0
                01AC 11581          VBUSDATA        <^X25>,0
            02  01AD 11582          .BYTE   ^X2                             ;CHMU
                01AE 11583          VBUSDATA        <^X24>,0
                01AF 11584          VBUSDATA        <^X25>,0
            02  01B0 11585          .BYTE   ^X2
                01B1 11586          VBUSDATA        <^X24>,0
                01B2 11587          VBUSDATA        <^X25>,0
            02  01B3 11588          .BYTE   ^X2
                01B4 11589          VBUSDATA        <^X24>,0
                01B5 11590          VBUSDATA        <^X25>,0
            02  01B6 11591          .BYTE   ^X2
                01B7 11592          VBUSDATA        <^X24>,0
                01B8 11593          VBUSDATA        <^X25>,0
            02  01B9 11594          .BYTE   ^X2
                01BA 11595          VBUSDATA        <^X24>,0
                01BB 11596          VBUSDATA        <^X25>,0
            02  01BC 11597          .BYTE   ^X2                             ;ADDL2
                01BD 11598          VBUSDATA        <^X24>,0
                01BE 11599          VBUSDATA        <^X25>,1
            02  01BF 11600          .BYTE   ^X2
                01C0 11601          VBUSDATA        <^X24>,0
                01C1 11602          VBUSDATA        <^X25>,0
```

```
                    02  01C2 11603          .BYTE   ^X2
                        01C3 11604          VBUSDATA        <^X24>,0
                        01C4 11605          VBUSDATA        <^X25>,0
                    02  01C5 11606          .BYTE   ^X2
                        01C6 11607          VBUSDATA        <^X24>,0
                        01C7 11608          VBUSDATA        <^X25>,0
                    02  01C8 11609          .BYTE   ^X2
                        01C9 11610          VBUSDATA        <^X24>,0
                        01CA 11611          VBUSDATA        <^X25>,0
                    02  01CB 11612          .BYTE   ^X2                             ;ADDL3
                        01CC 11613          VBUSDATA        <^X24>,0
                        01CD 11614          VBUSDATA        <^X25>,1
                    02  01CE 11615          .BYTE   ^X2
                        01CF 11616          VBUSDATA        <^X24>,0
                        01D0 11617          VBUSDATA        <^X25>,1
                    02  01D1 11618          .BYTE   ^X2
                        01D2 11619          VBUSDATA        <^X24>,0
                        01D3 11620          VBUSDATA        <^X25>,0
                    02  01D4 11621          .BYTE   ^X2
                        01D5 11622          VBUSDATA        <^X24>,0
                        01D6 11623          VBUSDATA        <^X25>,0
                    02  01D7 11624          .BYTE   ^X2
                        01D8 11625          VBUSDATA        <^X24>,0
                        01D9 11626          VBUSDATA        <^X25>,0
                    02  01DA 11627          .BYTE   ^X2                             ;SUBL2
                        01DB 11628          VBUSDATA        <^X24>,0
                        01DC 11629          VBUSDATA        <^X25>,1
                    02  01DD 11630          .BYTE   ^X2
                        01DE 11631          VBUSDATA        <^X24>,0
                        01DF 11632          VBUSDATA        <^X25>,0
                    02  01E0 11633          .BYTE   ^X2
                        01E1 11634          VBUSDATA        <^X24>,0
                        01E2 11635          VBUSDATA        <^X25>,0
                    02  01E3 11636          .BYTE   ^X2
                        01E4 11637          VBUSDATA        <^X24>,0
                        01E5 11638          VBUSDATA        <^X25>,0
                    02  01E6 11639          .BYTE   ^X2
                        01E7 11640          VBUSDATA        <^X24>,0
                        01E8 11641          VBUSDATA        <^X25>,0
                    02  01E9 11642          .BYTE   ^X2                             ;SUBL3
                        01EA 11643          VBUSDATA        <^X24>,0
                        01EB 11644          VBUSDATA        <^X25>,1
                    02  01EC 11645          .BYTE   ^X2
                        01ED 11646          VBUSDATA        <^X24>,0
                        01EE 11647          VBUSDATA        <^X25>,1
                    02  01EF 11648          .BYTE   ^X2
                        01F0 11649          VBUSDATA        <^X24>,0
                        01F1 11650          VBUSDATA        <^X25>,0
                    02  01F2 11651          .BYTE   ^X2
                        01F3 11652          VBUSDATA        <^X24>,0
                        01F4 11653          VBUSDATA        <^X25>,0
                    02  01F5 11654          .BYTE   ^X2
                        01F6 11655          VBUSDATA        <^X24>,0
                        01F7 11656          VBUSDATA        <^X25>,0
                    02  01F8 11657          .BYTE   ^X2                             ;MULL2
```

```
                    01F9 11658          VBUSDATA        <^X24>,0
                    01FA 11659          VBUSDATA        <^X25>,1
                 02 01FB 11660          .BYTE   ^X2
                    01FC 11661          VBUSDATA        <^X24>,0
                    01FD 11662          VBUSDATA        <^X25>,0
                 02 01FE 11663          .BYTE   ^X2
                    01FF 11664          VBUSDATA        <^X24>,0
                    0200 11665          VBUSDATA        <^X25>,0
                 02 0201 11666          .BYTE   ^X2
                    0202 11667          VBUSDATA        <^X24>,0
                    0203 11668          VBUSDATA        <^X25>,0
                 02 0204 11669          .BYTE   ^X2
                    0205 11670          VBUSDATA        <^X24>,0
                    0206 11671          VBUSDATA        <^X25>,0
                 02 0207 11672          .BYTE   ^X2                             ;MULL3
                    0208 11673          VBUSDATA        <^X24>,0
                    0209 11674          VBUSDATA        <^X25>,1
                 02 020A 11675          .BYTE   ^X2
                    020B 11676          VBUSDATA        <^X24>,0
                    020C 11677          VBUSDATA        <^X25>,1
                 02 020D 11678          .BYTE   ^X2
                    020E 11679          VBUSDATA        <^X24>,0
                    020F 11680          VBUSDATA        <^X25>,0
                 02 0210 11681          .BYTE   ^X2
                    0211 11682          VBUSDATA        <^X24>,0
                    0212 11683          VBUSDATA        <^X25>,0
                 02 0213 11684          .BYTE   ^X2
                    0214 11685          VBUSDATA        <^X24>,0
                    0215 11686          VBUSDATA        <^X25>,0
                 02 0216 11687          .BYTE   ^X2                             ;DIVL2
                    0217 11688          VBUSDATA        <^X24>,0
                    0218 11689          VBUSDATA        <^X25>,1
                 02 0219 11690          .BYTE   ^X2
                    021A 11691          VBUSDATA        <^X24>,0
                    021B 11692          VBUSDATA        <^X25>,0
                 02 021C 11693          .BYTE   ^X2
                    021D 11694          VBUSDATA        <^X24>,0
                    021E 11695          VBUSDATA        <^X25>,0
                 02 021F 11696          .BYTE   ^X2
                    0220 11697          VBUSDATA        <^X24>,0
                    0221 11698          VBUSDATA        <^X25>,0
                 02 0222 11699          .BYTE   ^X2
                    0223 11700          VBUSDATA        <^X24>,0
                    0224 11701          VBUSDATA        <^X25>,0
                 02 0225 11702          .BYTE   ^X2                             ;DIVL3
                    0226 11703          VBUSDATA        <^X24>,0
                    0227 11704          VBUSDATA        <^X25>,1
                 02 0228 11705          .BYTE   ^X2
                    0229 11706          VBUSDATA        <^X24>,0
                    022A 11707          VBUSDATA        <^X25>,1
                 02 022B 11708          .BYTE   ^X2
                    022C 11709          VBUSDATA        <^X24>,0
                    022D 11710          VBUSDATA        <^X25>,0
                 02 022E 11711          .BYTE   ^X2
                    022F 11712          VBUSDATA        <^X24>,0
```

```
                                  0230 11713          VBUSDATA        <^X25>,0
                              02  0231 11714          .BYTE   ^X2
                                  0232 11715          VBUSDATA        <^X24>,0
                                  0233 11716          VBUSDATA        <^X25>,0
                              02  0234 11717          .BYTE   ^X2                                  ;BISL2
                                  0235 11718          VBUSDATA        <^X24>,0
                                  0236 11719          VBUSDATA        <^X25>,1
                              02  0237 11720          .BYTE   ^X2
                                  0238 11721          VBUSDATA        <^X24>,0
                                  0239 11722          VBUSDATA        <^X25>,0
                              02  023A 11723          .BYTE   ^X2
                                  023B 11724          VBUSDATA        <^X24>,0
                                  023C 11725          VBUSDATA        <^X25>,0
                              02  023D 11726          .BYTE   ^X2
                                  023E 11727          VBUSDATA        <^X24>,0
                                  023F 11728          VBUSDATA        <^X25>,0
                              02  0240 11729          .BYTE   ^X2
                                  0241 11730          VBUSDATA        <^X24>,0
                                  0242 11731          VBUSDATA        <^X25>,0
                                  0243 11732
                                  0243 11733
                                  0243 11734 BEGIN_CPU_DATA
                                  0002 11735
                                  0002 11736 DCS_DATA
                                  0001 11737
U 00, 7700,C800,0364,0300,0470,1801  0001 11738 ;% 00   NOP
U 01, 7700,4800,5BE4,0300,2470,1802  000C 11742 ;% 01   WCTRL/LOADCRAR,WB_R[TEMP0]                ;DISABLE ^P INTERRUPT
U 02, 7700,0800,5BE4,0304,2C70,1803  0017 11746 ;% 02   WCTRL/CONWRITE,WB_R[TEMP1]                ;...
U 03, 7700,9800,D370,030F,FA70,1804  0022 11750 ;% 03   IPL_[1F]                                   ;SET IPL TO 1F
U 04, 5700,4840,0364,0308,0470,1805  002D 11754 ;% 04   R[TEMP2]_RDM                               ;MOVE LOOP COUNT TO RTEMP2
U 05, 7700,D800,EF64,0301,0470,1806  0038 11758 ;% 05   PL_[2]                                     ;SET PLATCH TO 2
U 06, 7700,C800,8B71,0308,0470,1807  0043 11762 ;% 06   Q_R[TEMP2].RL.P                            ;ROTATE LOOP COUNT LEFT 2
U 07, 7700,4800,03A4,03D8,4870,1808  004E 11766 ;% 07   WCTRL/PC_WB,WB_Q                           ;FORCE PREFETCH TO HAPPEN
U 08, 7700,4800,0364,1700,0470,1809  0059 11770 ;% 08   IRD1TEST                                   ;LOAD IR REGISTER
U 09, 7600,4800,0364,0300,0470,180A  0064 11774 ;% 09   DTYPE/IDEP,STROBE.V.BUS                    ;LOAD DSIZE INTO VBUS
U 0A, 7300,C800,0364,0300,0470,180B  006F 11778 ;% 0A   NOP,STOP.CLOCK                             ;STOP THE CPU CLOCK
U 0B, 7700,4800,0364,0300,0470,180C  007A 11782 ;% 0B   NOP
U 0C, 7700,C800,0364,1B00,0470,180D  0085 11786 ;% 0C   LOD INC BRA?                               ;INCREMENT THE IRD COUNTER
U 0D, 7600,C800,0364,0300,0470,180E  0090 11790 ;% 0D   DTYPE/IDEP,STROBE.V.BUS                    ;LOAD DSIZE INTO VBUS
U 0E, 7300,4800,0364,0300,0470,180F  009B 11794 ;% 0E   NOP,STOP.CLOCK                             ;STOP THE CPU CLOCK
                                  00A6 11798
                                  00A6 11799 RTEMP_DATA
                                  0001 11800
                          00C00000  0001 11801          .LONG   ^X00C00000                         ;DISABLE ^P INTERRUPT
                          00400000  0005 11802          .LONG   ^X00400000                         ;...
                                  0009 11803
                                  0009 11804 CACHE_DATA
                                  0001 11805
                          000000B0  0001 11806          .LONG   ^X000000B0                         ;MOVW
                          000000B1  0005 11807          .LONG   ^X000000B1                         ;CMPW
                          000000B2  0009 11808          .LONG   ^X000000B2                         ;MCOMW
                          000000B3  000D 11809          .LONG   ^X000000B3                         ;BITW
                          000000B4  0011 11810          .LONG   ^X000000B4                         ;CLRW
                          000000B5  0015 11811          .LONG   ^X000000B5                         ;TSTW
                          000000B6  0019 11812          .LONG   ^X000000B6                         ;INCW
```

```
000000B7  001D 11813          .LONG   ^X000000B7                      ;DECW
000000B8  0021 11814          .LONG   ^X000000B8                      ;BISPSW
000000B9  0025 11815          .LONG   ^X000000B9                      ;BICPSW
000000BA  0029 11816          .LONG   ^X000000BA                      ;POPR
000000BB  002D 11817          .LONG   ^X000000BB                      ;PUSHR
000000BC  0031 11818          .LONG   ^X000000BC                      ;CHMK
000000BD  0035 11819          .LONG   ^X000000BD                      ;CHME
000000BE  0039 11820          .LONG   ^X000000BE                      ;CHMS
000000BF  003D 11821          .LONG   ^X000000BF                      ;CHMU
000000C0  0041 11822          .LONG   ^X000000C0                      ;ADDL2
000000C1  0045 11823          .LONG   ^X000000C1                      ;ADDL3
000000C2  0049 11824          .LONG   ^X000000C2                      ;SUBL2
000000C3  004D 11825          .LONG   ^X000000C3                      ;SUBL3
000000C4  0051 11826          .LONG   ^X000000C4                      ;MULL2
000000C5  0055 11827          .LONG   ^X000000C5                      ;MULL3
000000C6  0059 11828          .LONG   ^X000000C6                      ;DIVL2
000000C7  005D 11829          .LONG   ^X000000C7                      ;DIVL3
000000C8  0061 11830          .LONG   ^X000000C8                      ;BISL2
00000000  0065 11831          .LONG   ^X00000000
          0069 11832
          0069 11833 END_CPU_DATA
          0243 11834
          0243 11835
          0243 11836 END_TEST_DATA
          0243
          0243        ;-----------------------------------------------------------------------------
          0243 11837 ENDTEST
```

```
                        0022        .SBTTL  "TEST   08C     NATIVE MODE DSIZE ROM TEST 10
                        0022 11839 ;*******************************************************************************
                        0022 11840 ;++
                        0022 11841 ;
                        0022 11842 ; FUNCTIONAL DESCRIPTION:
                        0022 11843 ;
                        0022 11844 ;       THIS IS ONE IN A SERIES OF 14 TESTS THAT CHECK THE CONTENTS OF THE
                        0022 11845 ;       DSIZE ROM.
                        0022 11846 ;
                        0022 11847 ;
                        0022 11848 ; TEST ALGORITHM:
                        0022 11849 ;
                        0022 11850 ;       1.  CREATE AN INDEX (K) TO INDEX RESULTS TABLE
                        0022 11851 ;       2.  CREATE LOOP (I) OF 25
                        0022 11852 ;       3.  SAVE LOOP COUNT IN 1$
                        0022 11853 ;       4.  MOVE LOOP COUNT TO WDREG
                        0022 11854 ;       5.  EXECUTE DCS PROGRAM
                        0022 11855 ;               a.  DISABLE ^P INTERRUPTS
                        0022 11856 ;               b.  SET IPL TO 1F
                        0022 11857 ;               c.  MOVE LOOP COUNT TO RTEMP2
                        0022 11858 ;               d.  ROTATE LOOP COUNT LEFT 2 TO FORM CACHE ADDRESS
                        0022 11859 ;               e.  LOAD THIS ADDRESS INTO PC TO CAUSE PREFETCH
                        0022 11860 ;               f.  PERFORM IRD1TEST TO CAUSE IR TO LOAD INSTRUCTION
                        0022 11861 ;               g.  SET DTYPE TO INSTURCTION DEPENDENT AND STROBE THE VBUS
                        0022 11862 ;       6.  TEST VBUS FOR CORRECT DSIZE
                        0022 11863 ;       7.  IF NO ERROR CREATE LOOP (J) OF 5
                        0022 11864 ;       8.  EXECUTE DCS PROGRAM
                        0022 11865 ;               a.  INCREMENT THE IRD COUNTER
                        0022 11866 ;               b.  SET DTYPE TO INSTRUCTION DEPENDENT AND STROBE THE VBUS
                        0022 11867 ;       9.  TEST VBUS FOR CORRECT DSIZE
                        0022 11868 ;      10.  INCREMENT INDEX K BY ONE
                        0022 11869 ;      11.  IF NO ERROR GO TO STEP 8 UNTIL LOOP J IS EXHAUSTED
                        0022 11870 ;      12.  GO TO STEP 3 UNTIL LOOP I IS EXHAUSTED
                        0022 11871 ;
                        0022 11872 ;
                        0022 11873 ; TEST PATTERNS:
                        0022 11874 ;
                        0022 11875 ;       LOOP COUNT I             INSTRUCTION
                        0022 11876 ;            1                      BISL3
                        0022 11877 ;            2                      BICL2
                        0022 11878 ;            3                      BICL3
                        0022 11879 ;            4                      XORL2
                        0022 11880 ;            5                      XORL3
                        0022 11881 ;            6                      MNEGL
                        0022 11882 ;            7                      CASEL
                        0022 11883 ;            8                      MOVL
                        0022 11884 ;            9                      CMPL
                        0022 11885 ;            A                      MCOML
                        0022 11886 ;            B                      BITL
                        0022 11887 ;            C                      CLRL
                        0022 11888 ;            D                      TSTL
                        0022 11889 ;            E                      INCL
                        0022 11890 ;            F                      DECL
                        0022 11891 ;           10                      ADWC
                        0022 11892 ;           11                      SBWC
```

```
                        0022 11893 ;           12                   MTPR
                        0022 11894 ;           13                   MFPR
                        0022 11895 ;           14                   MOVPSL
                        0022 11896 ;           15                   PUSHL
                        0022 11897 ;           16                   MOVAL
                        0022 11898 ;           17                   PUSHAL
                        0022 11899 ;           18                   BBS
                        0022 11900 ;           19                   BBC
                        0022 11901 ;
                        0022 11902 ;
                        0022 11903 ; LOGIC DESCRIPTION:
                        0022 11904 ;
                        0022 11905 ;       THIS TEST ALONG WITH THE OTHER 13 TESTS IN THIS SERIES CHECK ALL THE
                        0022 11906 ;       LOCATIONS BLASTED INTO THE DSIZE ROM.  ANY FALIURES SHOULD BE CONFINED
                        0022 11907 ;       TO THE ROM.
                        0022 11908 ;
                        0022 11909 ;
                        0022 11910 ; ASSUMPTIONS:
                        0022 11911 ;
                        0022 11912 ;       THIS TEST ASSUMES THE PREFETCH LOGIC AND INSTRUCTION REGISTER ARE
                        0022 11913 ;       OPERATIONAL.
                        0022 11914 ;
                        0022 11915 ;
                        0022 11916 ; ERROR DESCRIPTION:
                        0022 11917 ;
                        0022 11918 ;       EXPECTED VBUS DATA (DSIZE)
                        0022 11919 ;       RECEIVED VBUS DATA
                        0022 11920 ;       LOOP COUNT I
                        0022 11921 ;       LOOP COUNT J
                        0022 11922 ;       LOOP COUNT K
                        0022 11923 ;              (BITS <7:0> = VBUS BIT POSITION)
                        0022 11924 ;              (BITS <8> = BIT STATE)
                        0022 11925 ;
                        0022 11926 ;--
                        0022 11927 ;*******************************************************************************
                        0022 11928
                        0022 11929
                        0022 11930         INITIALIZE                              ;INIT THE CPU
                        0024 11931         LDINDEX         K,1                     ;INIT INDEX K TO 1
                        002B 11932         LOOP            I,1,25                  ;CREATE A TEST LOOP OF 50
                        0032 11933         SAVEINDEX       I,1$                    ;SAVE LOOP COUNT IN 1$
                        0037 11934         LOADREG         WDREG,1$,,L             ;LOAD LOOP COUNT INTO WDREG
                        003F 11935         ERRLOOP                                 ;ERROR LOOP POINT
                        0041 11936         FETCH           0                       ;START DCS PROGRAM
                        0046 11937         BRSTCLK         <^X0A>,INH              ;END DCS PROGRAM
                        004A 11938         CMPVBUS         2$,I                    ;TEST DSIZE
                        004F 11939         IFERROR                                 ;ROM FAILURE
                        0054 11940         LOOP            J,1,5                   ;CREATE TEST LOOP OF 5
                        005B 11941         FETCH           <^X0B>                  ;START DCS PROGRAM
                        0060 11942         BRSTCLK         <^X0E>,INH              ;END DCS PROGRAM
                        0064 11943         CMPVBUS         3$,K                    ;TEST DSIZE
                        0069 11944         IFERROR                                 ;ROM FAILURE
                        006E 11945         INCINDEX        K                       ;INCREMENT INDEX K BY ONE
                        0071 11946         ENDLOOP         J                       ;END TEST LOOP J
                        0074 11947         ENDLOOP         I                       ;END TEST LOOP I
```

```
                  0077 11948          SKIP                                        ;GO TO NEXT TEST
                  007E 11949
                  007E 11950
                  007E 11951 BEGIN_TEST_DATA
                  007E
                  007E       ;-----------------------------------------------------------------------
                  007E 11952
        00000000  007E 11953 1$:      .LONG   ^X00000000                          ;LOOP COUNT STORAGE
              02  0082 11954 2$:      .BYTE   ^X2                                 ;RESULTS TABLE (IRD 0)
                  0083 11955          VBUSDATA        <^X24>,0                    ;BISL3
                  0084 11956          VBUSDATA        <^X25>,1
              02  0085 11957          .BYTE   ^X2                                 ;BICL2
                  0086 11958          VBUSDATA        <^X24>,0
                  0087 11959          VBUSDATA        <^X25>,1
              02  0088 11960          .BYTE   ^X2                                 ;BICL3
                  0089 11961          VBUSDATA        <^X24>,0
                  008A 11962          VBUSDATA        <^X25>,1
              02  008B 11963          .BYTE   ^X2                                 ;XORL2
                  008C 11964          VBUSDATA        <^X24>,0
                  008D 11965          VBUSDATA        <^X25>,1
              02  008E 11966          .BYTE   ^X2                                 ;XORL3
                  008F 11967          VBUSDATA        <^X24>,0
                  0090 11968          VBUSDATA        <^X25>,1
              02  0091 11969          .BYTE   ^X2                                 ;MNEGL
                  0092 11970          VBUSDATA        <^X24>,0
                  0093 11971          VBUSDATA        <^X25>,1
              02  0094 11972          .BYTE   ^X2                                 ;CASEL
                  0095 11973          VBUSDATA        <^X24>,0
                  0096 11974          VBUSDATA        <^X25>,1
              02  0097 11975          .BYTE   ^X2                                 ;MOVL
                  0098 11976          VBUSDATA        <^X24>,0
                  0099 11977          VBUSDATA        <^X25>,1
              02  009A 11978          .BYTE   ^X2                                 ;CMPL
                  009B 11979          VBUSDATA        <^X24>,0
                  009C 11980          VBUSDATA        <^X25>,1
              02  009D 11981          .BYTE   ^X2                                 ;MCOML
                  009E 11982          VBUSDATA        <^X24>,0
                  009F 11983          VBUSDATA        <^X25>,1
              02  00A0 11984          .BYTE   ^X2                                 ;BITL
                  00A1 11985          VBUSDATA        <^X24>,0
                  00A2 11986          VBUSDATA        <^X25>,1
              02  00A3 11987          .BYTE   ^X2                                 ;CLRL
                  00A4 11988          VBUSDATA        <^X24>,0
                  00A5 11989          VBUSDATA        <^X25>,1
              02  00A6 11990          .BYTE   ^X2                                 ;TSTL
                  00A7 11991          VBUSDATA        <^X24>,0
                  00A8 11992          VBUSDATA        <^X25>,1
              02  00A9 11993          .BYTE   ^X2                                 ;INCL
                  00AA 11994          VBUSDATA        <^X24>,0
                  00AB 11995          VBUSDATA        <^X25>,1
              02  00AC 11996          .BYTE   ^X2                                 ;DECL
                  00AD 11997          VBUSDATA        <^X24>,0
                  00AE 11998          VBUSDATA        <^X25>,1
              02  00AF 11999          .BYTE   ^X2                                 ;ADWC
                  00B0 12000          VBUSDATA        <^X24>,0
```

```
                00B1 12001          VBUSDATA        <^X25>,1
            02  00B2 12002          .BYTE   ^X2                             ;SBWC
                00B3 12003          VBUSDATA        <^X24>,0
                00B4 12004          VBUSDATA        <^X25>,1
            02  00B5 12005          .BYTE   ^X2                             ;MTPR
                00B6 12006          VBUSDATA        <^X24>,0
                00B7 12007          VBUSDATA        <^X25>,1
            02  00B8 12008          .BYTE   ^X2                             ;MFPR
                00B9 12009          VBUSDATA        <^X24>,0
                00BA 12010          VBUSDATA        <^X25>,1
            02  00BB 12011          .BYTE   ^X2                             ;MOVPSL
                00BC 12012          VBUSDATA        <^X24>,0
                00BD 12013          VBUSDATA        <^X25>,1
            02  00BE 12014          .BYTE   ^X2                             ;PUSHL
                00BF 12015          VBUSDATA        <^X24>,0
                00C0 12016          VBUSDATA        <^X25>,1
            02  00C1 12017          .BYTE   ^X2                             ;MOVAL
                00C2 12018          VBUSDATA        <^X24>,0
                00C3 12019          VBUSDATA        <^X25>,1
            02  00C4 12020          .BYTE   ^X2                             ;PUSHAL
                00C5 12021          VBUSDATA        <^X24>,0
                00C6 12022          VBUSDATA        <^X25>,1
            02  00C7 12023          .BYTE   ^X2                             ;BBS
                00C8 12024          VBUSDATA        <^X24>,0
                00C9 12025          VBUSDATA        <^X25>,1
            02  00CA 12026          .BYTE   ^X2                             ;BBC
                00CB 12027          VBUSDATA        <^X24>,0
                00CC 12028          VBUSDATA        <^X25>,1
                00CD 12029
            02  00CD 12030 3$:      .BYTE   ^X2                             ;RESULTS TABLE (IRD 1 TO 5)
                00CE 12031          VBUSDATA        <^X24>,0                ;BISL3
                00CF 12032          VBUSDATA        <^X25>,1
            02  00D0 12033          .BYTE   ^X2
                00D1 12034          VBUSDATA        <^X24>,0
                00D2 12035          VBUSDATA        <^X25>,1
            02  00D3 12036          .BYTE   ^X2
                00D4 12037          VBUSDATA        <^X24>,0
                00D5 12038          VBUSDATA        <^X25>,0
            02  00D6 12039          .BYTE   ^X2
                00D7 12040          VBUSDATA        <^X24>,0
                00D8 12041          VBUSDATA        <^X25>,0
            02  00D9 12042          .BYTE   ^X2
                00DA 12043          VBUSDATA        <^X24>,0
                00DB 12044          VBUSDATA        <^X25>,0
            02  00DC 12045          .BYTE   ^X2                             ;BICL2
                00DD 12046          VBUSDATA        <^X24>,0
                00DE 12047          VBUSDATA        <^X25>,1
            02  00DF 12048          .BYTE   ^X2
                00E0 12049          VBUSDATA        <^X24>,0
                00E1 12050          VBUSDATA        <^X25>,0
            02  00E2 12051          .BYTE   ^X2
                00E3 12052          VBUSDATA        <^X24>,0
                00E4 12053          VBUSDATA        <^X25>,0
            02  00E5 12054          .BYTE   ^X2
                00E6 12055          VBUSDATA        <^X24>,0
```

```
                    00E7 12056          VBUSDATA        <^X25>,0
                02  00E8 12057          .BYTE   ^X2
                    00E9 12058          VBUSDATA        <^X24>,0
                    00EA 12059          VBUSDATA        <^X25>,0
                02  00EB 12060          .BYTE   ^X2                             ;BICL3
                    00EC 12061          VBUSDATA        <^X24>,0
                    00ED 12062          VBUSDATA        <^X25>,1
                02  00EE 12063          .BYTE   ^X2
                    00EF 12064          VBUSDATA        <^X24>,0
                    00F0 12065          VBUSDATA        <^X25>,1
                02  00F1 12066          .BYTE   ^X2
                    00F2 12067          VBUSDATA        <^X24>,0
                    00F3 12068          VBUSDATA        <^X25>,0
                02  00F4 12069          .BYTE   ^X2
                    00F5 12070          VBUSDATA        <^X24>,0
                    00F6 12071          VBUSDATA        <^X25>,0
                02  00F7 12072          .BYTE   ^X2
                    00F8 12073          VBUSDATA        <^X24>,0
                    00F9 12074          VBUSDATA        <^X25>,0
                02  00FA 12075          .BYTE   ^X2                             ;XORL2
                    00FB 12076          VBUSDATA        <^X24>,0
                    00FC 12077          VBUSDATA        <^X25>,1
                02  00FD 12078          .BYTE   ^X2
                    00FE 12079          VBUSDATA        <^X24>,0
                    00FF 12080          VBUSDATA        <^X25>,0
                02  0100 12081          .BYTE   ^X2
                    0101 12082          VBUSDATA        <^X24>,0
                    0102 12083          VBUSDATA        <^X25>,0
                02  0103 12084          .BYTE   ^X2
                    0104 12085          VBUSDATA        <^X24>,0
                    0105 12086          VBUSDATA        <^X25>,0
                02  0106 12087          .BYTE   ^X2
                    0107 12088          VBUSDATA        <^X24>,0
                    0108 12089          VBUSDATA        <^X25>,0
                02  0109 12090          .BYTE   ^X2                             ;XORL3
                    010A 12091          VBUSDATA        <^X24>,0
                    010B 12092          VBUSDATA        <^X25>,1
                02  010C 12093          .BYTE   ^X2
                    010D 12094          VBUSDATA        <^X24>,0
                    010E 12095          VBUSDATA        <^X25>,1
                02  010F 12096          .BYTE   ^X2
                    0110 12097          VBUSDATA        <^X24>,0
                    0111 12098          VBUSDATA        <^X25>,0
                02  0112 12099          .BYTE   ^X2
                    0113 12100          VBUSDATA        <^X24>,0
                    0114 12101          VBUSDATA        <^X25>,0
                02  0115 12102          .BYTE   ^X2
                    0116 12103          VBUSDATA        <^X24>,0
                    0117 12104          VBUSDATA        <^X25>,0
                02  0118 12105          .BYTE   ^X2                             ;MNEGL
                    0119 12106          VBUSDATA        <^X24>,0
                    011A 12107          VBUSDATA        <^X25>,1
                02  011B 12108          .BYTE   ^X2
                    011C 12109          VBUSDATA        <^X24>,0
                    011D 12110          VBUSDATA        <^X25>,0
```

```
                02  011E 12111          .BYTE   ^X2
                    011F 12112          VBUSDATA        <^X24>,0
                    0120 12113          VBUSDATA        <^X25>,0
                02  0121 12114          .BYTE   ^X2
                    0122 12115          VBUSDATA        <^X24>,0
                    0123 12116          VBUSDATA        <^X25>,0
                02  0124 12117          .BYTE   ^X2
                    0125 12118          VBUSDATA        <^X24>,0
                    0126 12119          VBUSDATA        <^X25>,0
                02  0127 12120          .BYTE   ^X2                             ;CASEL
                    0128 12121          VBUSDATA        <^X24>,0
                    0129 12122          VBUSDATA        <^X25>,1
                02  012A 12123          .BYTE   ^X2
                    012B 12124          VBUSDATA        <^X24>,0
                    012C 12125          VBUSDATA        <^X25>,1
                02  012D 12126          .BYTE   ^X2
                    012E 12127          VBUSDATA        <^X24>,0
                    012F 12128          VBUSDATA        <^X25>,0
                02  0130 12129          .BYTE   ^X2
                    0131 12130          VBUSDATA        <^X24>,0
                    0132 12131          VBUSDATA        <^X25>,0
                02  0133 12132          .BYTE   ^X2
                    0134 12133          VBUSDATA        <^X24>,0
                    0135 12134          VBUSDATA        <^X25>,0
                02  0136 12135          .BYTE   ^X2                             ;MOVL
                    0137 12136          VBUSDATA        <^X24>,0
                    0138 12137          VBUSDATA        <^X25>,1
                02  0139 12138          .BYTE   ^X2
                    013A 12139          VBUSDATA        <^X24>,0
                    013B 12140          VBUSDATA        <^X25>,0
                02  013C 12141          .BYTE   ^X2
                    013D 12142          VBUSDATA        <^X24>,0
                    013E 12143          VBUSDATA        <^X25>,0
                02  013F 12144          .BYTE   ^X2
                    0140 12145          VBUSDATA        <^X24>,0
                    0141 12146          VBUSDATA        <^X25>,0
                02  0142 12147          .BYTE   ^X2
                    0143 12148          VBUSDATA        <^X24>,0
                    0144 12149          VBUSDATA        <^X25>,0
                02  0145 12150          .BYTE   ^X2                             ;CMPL
                    0146 12151          VBUSDATA        <^X24>,0
                    0147 12152          VBUSDATA        <^X25>,1
                02  0148 12153          .BYTE   ^X2
                    0149 12154          VBUSDATA        <^X24>,0
                    014A 12155          VBUSDATA        <^X25>,0
                02  014B 12156          .BYTE   ^X2
                    014C 12157          VBUSDATA        <^X24>,0
                    014D 12158          VBUSDATA        <^X25>,0
                02  014E 12159          .BYTE   ^X2
                    014F 12160          VBUSDATA        <^X24>,0
                    0150 12161          VBUSDATA        <^X25>,0
                02  0151 12162          .BYTE   ^X2
                    0152 12163          VBUSDATA        <^X24>,0
                    0153 12164          VBUSDATA        <^X25>,0
                02  0154 12165          .BYTE   ^X2                             ;MCOML
```

```
                0155 12166          VBUSDATA        <^X24>,0
                0156 12167          VBUSDATA        <^X25>,1
          02    0157 12168          .BYTE   ^X2
                0158 12169          VBUSDATA        <^X24>,0
                0159 12170          VBUSDATA        <^X25>,0
          02    015A 12171          .BYTE   ^X2
                015B 12172          VBUSDATA        <^X24>,0
                015C 12173          VBUSDATA        <^X25>,0
          02    015D 12174          .BYTE   ^X2
                015E 12175          VBUSDATA        <^X24>,0
                015F 12176          VBUSDATA        <^X25>,0
          02    0160 12177          .BYTE   ^X2
                0161 12178          VBUSDATA        <^X24>,0
                0162 12179          VBUSDATA        <^X25>,0
          02    0163 12180          .BYTE   ^X2                             ;BITL
                0164 12181          VBUSDATA        <^X24>,0
                0165 12182          VBUSDATA        <^X25>,1
          02    0166 12183          .BYTE   ^X2
                0167 12184          VBUSDATA        <^X24>,0
                0168 12185          VBUSDATA        <^X25>,0
          02    0169 12186          .BYTE   ^X2
                016A 12187          VBUSDATA        <^X24>,0
                016B 12188          VBUSDATA        <^X25>,0
          02    016C 12189          .BYTE   ^X2
                016D 12190          VBUSDATA        <^X24>,0
                016E 12191          VBUSDATA        <^X25>,0
          02    016F 12192          .BYTE   ^X2
                0170 12193          VBUSDATA        <^X24>,0
                0171 12194          VBUSDATA        <^X25>,0
          02    0172 12195          .BYTE   ^X2                             ;CLRL
                0173 12196          VBUSDATA        <^X24>,0
                0174 12197          VBUSDATA        <^X25>,0
          02    0175 12198          .BYTE   ^X2
                0176 12199          VBUSDATA        <^X24>,0
                0177 12200          VBUSDATA        <^X25>,0
          02    0178 12201          .BYTE   ^X2
                0179 12202          VBUSDATA        <^X24>,0
                017A 12203          VBUSDATA        <^X25>,0
          02    017B 12204          .BYTE   ^X2
                017C 12205          VBUSDATA        <^X24>,0
                017D 12206          VBUSDATA        <^X25>,0
          02    017E 12207          .BYTE   ^X2
                017F 12208          VBUSDATA        <^X24>,0
                0180 12209          VBUSDATA        <^X25>,0
          02    0181 12210          .BYTE   ^X2                             ;TSTL
                0182 12211          VBUSDATA        <^X24>,0
                0183 12212          VBUSDATA        <^X25>,0
          02    0184 12213          .BYTE   ^X2
                0185 12214          VBUSDATA        <^X24>,0
                0186 12215          VBUSDATA        <^X25>,0
          02    0187 12216          .BYTE   ^X2
                0188 12217          VBUSDATA        <^X24>,0
                0189 12218          VBUSDATA        <^X25>,0
          02    018A 12219          .BYTE   ^X2
                018B 12220          VBUSDATA        <^X24>,0
```

```
                018C 12221          VBUSDATA        <^X25>,0
            02  018D 12222          .BYTE   ^X2
                018E 12223          VBUSDATA        <^X24>,0
                018F 12224          VBUSDATA        <^X25>,0
            02  0190 12225          .BYTE   ^X2                             ;INCL
                0191 12226          VBUSDATA        <^X24>,0
                0192 12227          VBUSDATA        <^X25>,0
            02  0193 12228          .BYTE   ^X2
                0194 12229          VBUSDATA        <^X24>,0
                0195 12230          VBUSDATA        <^X25>,0
            02  0196 12231          .BYTE   ^X2
                0197 12232          VBUSDATA        <^X24>,0
                0198 12233          VBUSDATA        <^X25>,0
            02  0199 12234          .BYTE   ^X2
                019A 12235          VBUSDATA        <^X24>,0
                019B 12236          VBUSDATA        <^X25>,0
            02  019C 12237          .BYTE   ^X2
                019D 12238          VBUSDATA        <^X24>,0
                019E 12239          VBUSDATA        <^X25>,0
            02  019F 12240          .BYTE   ^X2                             ;DECL
                01A0 12241          VBUSDATA        <^X24>,0
                01A1 12242          VBUSDATA        <^X25>,0
            02  01A2 12243          .BYTE   ^X2
                01A3 12244          VBUSDATA        <^X24>,0
                01A4 12245          VBUSDATA        <^X25>,0
            02  01A5 12246          .BYTE   ^X2
                01A6 12247          VBUSDATA        <^X24>,0
                01A7 12248          VBUSDATA        <^X25>,0
            02  01A8 12249          .BYTE   ^X2
                01A9 12250          VBUSDATA        <^X24>,0
                01AA 12251          VBUSDATA        <^X25>,0
            02  01AB 12252          .BYTE   ^X2
                01AC 12253          VBUSDATA        <^X24>,0
                01AD 12254          VBUSDATA        <^X25>,0
            02  01AE 12255          .BYTE   ^X2                             ;ADWC
                01AF 12256          VBUSDATA        <^X24>,0
                01B0 12257          VBUSDATA        <^X25>,1
            02  01B1 12258          .BYTE   ^X2
                01B2 12259          VBUSDATA        <^X24>,0
                01B3 12260          VBUSDATA        <^X25>,0
            02  01B4 12261          .BYTE   ^X2
                01B5 12262          VBUSDATA        <^X24>,0
                01B6 12263          VBUSDATA        <^X25>,0
            02  01B7 12264          .BYTE   ^X2
                01B8 12265          VBUSDATA        <^X24>,0
                01B9 12266          VBUSDATA        <^X25>,0
            02  01BA 12267          .BYTE   ^X2
                01BB 12268          VBUSDATA        <^X24>,0
                01BC 12269          VBUSDATA        <^X25>,0
            02  01BD 12270          .BYTE   ^X2                             ;SBWC
                01BE 12271          VBUSDATA        <^X24>,0
                01BF 12272          VBUSDATA        <^X25>,1
            02  01C0 12273          .BYTE   ^X2
                01C1 12274          VBUSDATA        <^X24>,0
                01C2 12275          VBUSDATA        <^X25>,0
```

```
                    02  01C3 12276          .BYTE   ^X2
                        01C4 12277          VBUSDATA        <^X24>,0
                        01C5 12278          VBUSDATA        <^X25>,0
                    02  01C6 12279          .BYTE   ^X2
                        01C7 12280          VBUSDATA        <^X24>,0
                        01C8 12281          VBUSDATA        <^X25>,0
                    02  01C9 12282          .BYTE   ^X2
                        01CA 12283          VBUSDATA        <^X24>,0
                        01CB 12284          VBUSDATA        <^X25>,0
                    02  01CC 12285          .BYTE   ^X2                             ;MTPR
                        01CD 12286          VBUSDATA        <^X24>,0
                        01CE 12287          VBUSDATA        <^X25>,1
                    02  01CF 12288          .BYTE   ^X2
                        01D0 12289          VBUSDATA        <^X24>,0
                        01D1 12290          VBUSDATA        <^X25>,0
                    02  01D2 12291          .BYTE   ^X2
                        01D3 12292          VBUSDATA        <^X24>,0
                        01D4 12293          VBUSDATA        <^X25>,0
                    02  01D5 12294          .BYTE   ^X2
                        01D6 12295          VBUSDATA        <^X24>,0
                        01D7 12296          VBUSDATA        <^X25>,0
                    02  01D8 12297          .BYTE   ^X2
                        01D9 12298          VBUSDATA        <^X24>,0
                        01DA 12299          VBUSDATA        <^X25>,0
                    02  01DB 12300          .BYTE   ^X2                             ;MFPR
                        01DC 12301          VBUSDATA        <^X24>,0
                        01DD 12302          VBUSDATA        <^X25>,1
                    02  01DE 12303          .BYTE   ^X2
                        01DF 12304          VBUSDATA        <^X24>,0
                        01E0 12305          VBUSDATA        <^X25>,0
                    02  01E1 12306          .BYTE   ^X2
                        01E2 12307          VBUSDATA        <^X24>,0
                        01E3 12308          VBUSDATA        <^X25>,0
                    02  01E4 12309          .BYTE   ^X2
                        01E5 12310          VBUSDATA        <^X24>,0
                        01E6 12311          VBUSDATA        <^X25>,0
                    02  01E7 12312          .BYTE   ^X2
                        01E8 12313          VBUSDATA        <^X24>,0
                        01E9 12314          VBUSDATA        <^X25>,0
                    02  01EA 12315          .BYTE   ^X2                             ;MOVPSL
                        01EB 12316          VBUSDATA        <^X24>,0
                        01EC 12317          VBUSDATA        <^X25>,0
                    02  01ED 12318          .BYTE   ^X2
                        01EE 12319          VBUSDATA        <^X24>,0
                        01EF 12320          VBUSDATA        <^X25>,0
                    02  01F0 12321          .BYTE   ^X2
                        01F1 12322          VBUSDATA        <^X24>,0
                        01F2 12323          VBUSDATA        <^X25>,0
                    02  01F3 12324          .BYTE   ^X2
                        01F4 12325          VBUSDATA        <^X24>,0
                        01F5 12326          VBUSDATA        <^X25>,0
                    02  01F6 12327          .BYTE   ^X2
                        01F7 12328          VBUSDATA        <^X24>,0
                        01F8 12329          VBUSDATA        <^X25>,0
                    02  01F9 12330          .BYTE   ^X2                             ;PUSHL
```

```
                    01FA 12331          VBUSDATA        <^X24>,0
                    01FB 12332          VBUSDATA        <^X25>,0
                 02 01FC 12333          .BYTE   ^X2
                    01FD 12334          VBUSDATA        <^X24>,0
                    01FE 12335          VBUSDATA        <^X25>,0
                 02 01FF 12336          .BYTE   ^X2
                    0200 12337          VBUSDATA        <^X24>,0
                    0201 12338          VBUSDATA        <^X25>,0
                 02 0202 12339          .BYTE   ^X2
                    0203 12340          VBUSDATA        <^X24>,0
                    0204 12341          VBUSDATA        <^X25>,0
                 02 0205 12342          .BYTE   ^X2
                    0206 12343          VBUSDATA        <^X24>,0
                    0207 12344          VBUSDATA        <^X25>,0
                 02 0208 12345          .BYTE   ^X2                             ;MOVAL
                    0209 12346          VBUSDATA        <^X24>,0
                    020A 12347          VBUSDATA        <^X25>,1
                 02 020B 12348          .BYTE   ^X2
                    020C 12349          VBUSDATA        <^X24>,0
                    020D 12350          VBUSDATA        <^X25>,0
                 02 020E 12351          .BYTE   ^X2
                    020F 12352          VBUSDATA        <^X24>,0
                    0210 12353          VBUSDATA        <^X25>,0
                 02 0211 12354          .BYTE   ^X2
                    0212 12355          VBUSDATA        <^X24>,0
                    0213 12356          VBUSDATA        <^X25>,0
                 02 0214 12357          .BYTE   ^X2
                    0215 12358          VBUSDATA        <^X24>,0
                    0216 12359          VBUSDATA        <^X25>,0
                 02 0217 12360          .BYTE   ^X2                             ;PUSHAL
                    0218 12361          VBUSDATA        <^X24>,0
                    0219 12362          VBUSDATA        <^X25>,0
                 02 021A 12363          .BYTE   ^X2
                    021B 12364          VBUSDATA        <^X24>,0
                    021C 12365          VBUSDATA        <^X25>,0
                 02 021D 12366          .BYTE   ^X2
                    021E 12367          VBUSDATA        <^X24>,0
                    021F 12368          VBUSDATA        <^X25>,0
                 02 0220 12369          .BYTE   ^X2
                    0221 12370          VBUSDATA        <^X24>,0
                    0222 12371          VBUSDATA        <^X25>,0
                 02 0223 12372          .BYTE   ^X2
                    0224 12373          VBUSDATA        <^X24>,0
                    0225 12374          VBUSDATA        <^X25>,0
                 02 0226 12375          .BYTE   ^X2                             ;BBS
                    0227 12376          VBUSDATA        <^X24>,0
                    0228 12377          VBUSDATA        <^X25>,0
                 02 0229 12378          .BYTE   ^X2
                    022A 12379          VBUSDATA        <^X24>,0
                    022B 12380          VBUSDATA        <^X25>,0
                 02 022C 12381          .BYTE   ^X2
                    022D 12382          VBUSDATA        <^X24>,0
                    022E 12383          VBUSDATA        <^X25>,0
                 02 022F 12384          .BYTE   ^X2
                    0230 12385          VBUSDATA        <^X24>,0
```

```
                                0231 12386          VBUSDATA        <^X25>,0
                             02 0232 12387          .BYTE   ^X2
                                0233 12388          VBUSDATA        <^X24>,0
                                0234 12389          VBUSDATA        <^X25>,0
                             02 0235 12390          .BYTE   ^X2                             ;BBC
                                0236 12391          VBUSDATA        <^X24>,0
                                0237 12392          VBUSDATA        <^X25>,0
                             02 0238 12393          .BYTE   ^X2
                                0239 12394          VBUSDATA        <^X24>,0
                                023A 12395          VBUSDATA        <^X25>,0
                             02 023B 12396          .BYTE   ^X2
                                023C 12397          VBUSDATA        <^X24>,0
                                023D 12398          VBUSDATA        <^X25>,0
                             02 023E 12399          .BYTE   ^X2
                                023F 12400          VBUSDATA        <^X24>,0
                                0240 12401          VBUSDATA        <^X25>,0
                             02 0241 12402          .BYTE   ^X2
                                0242 12403          VBUSDATA        <^X24>,0
                                0243 12404          VBUSDATA        <^X25>,0
                                0244 12405
                                0244 12406 BEGIN_CPU_DATA
                                0002 12407
                                0002 12408 DCS_DATA
                                0001 12409
U 00, 7700,C800,0364,0300,0470,1801 0001 12410 ;* 00   NOP
U 01, 7700,4800,5BE4,0300,2470,1802 000C 12414 ;* 01   WCTRL/LOADCRAR,WB_R[TEMP0]           ;DISABLE ^P INTERRUPT
U 02, 7700,0800,5BE4,0304,2C70,1803 0017 12418 ;* 02   WCTRL/CONWRITE,WB_R[TEMP1]          ;...
U 03, 7700,9800,D370,030F,FA70,1804 0022 12422 ;* 03   IPL_[1F]                            ;SET IPL TO 1F
U 04, 5700,4840,0364,0308,0470,1805 002D 12426 ;* 04   R[TEMP2]_RDM                        ;MOVE LOOP COUNT TO RTEMP2
U 05, 7700,D800,EF64,0301,0470,1806 0038 12430 ;* 05   PL_[2]                             ;SET PLATCH TO 2
U 06, 7700,C800,8B71,0308,0470,1807 0043 12434 ;* 06   Q_R[TEMP2].RL.P                    ;ROTATE LOOP COUNT LEFT 2
U 07, 7700,4800,03A4,03D8,4870,1808 004E 12438 ;* 07   WCTRL/PC_WB,WB_Q                    ;FORCE PREFETCH TO HAPPEN
U 08, 7700,4800,0364,1700,0470,1809 0059 12442 ;* 08   IRD1TEST                           ;LOAD IR REGISTER
U 09, 7600,4800,0364,0300,0470,180A 0064 12446 ;* 09   DTYPE/IDEP,STROBE.V.BUS            ;LOAD DSIZE INTO VBUS
U 0A, 7300,C800,0364,0300,0470,180B 006F 12450 ;* 0A   NOP,STOP.CLOCK                     ;STOP THE CPU CLOCK
U 0B, 7700,4800,0364,0300,0470,180C 007A 12454 ;* 0B   NOP
U 0C, 7700,C800,0364,1B00,0470,180D 0085 12458 ;* 0C   LOD INC BRA?                       ;INCREMENT THE IRD COUNTER
U 0D, 7600,C800,0364,0300,0470,180E 0090 12462 ;* 0D   DTYPE/IDEP,STROBE.V.BUS            ;LOAD DSIZE INTO VBUS
U 0E, 7300,4800,0364,0300,0470,180F 009B 12466 ;* 0E   NOP,STOP.CLOCK                     ;STOP THE CPU CLOCK
                                00A6 12470
                                00A6 12471 RTEMP_DATA
                                0001 12472
                          00C00000 0001 12473          .LONG   ^X00C00000                ;DISABLE ^P INTERRUPT
                          00400000 0005 12474          .LONG   ^X00400000                ;...
                                0009 12475
                                0009 12476 CACHE_DATA
                                0001 12477
                          000000C9 0001 12478          .LONG   ^X000000C9                ;BISL3
                          000000CA 0005 12479          .LONG   ^X000000CA                ;BICL2
                          000000CB 0009 12480          .LONG   ^X000000CB                ;BICL3
                          000000CC 000D 12481          .LONG   ^X000000CC                ;XORL2
                          000000CD 0011 12482          .LONG   ^X000000CD                ;XORL3
                          000000CE 0015 12483          .LONG   ^X000000CE                ;MNEGL
                          000000CF 0019 12484          .LONG   ^X000000CF                ;CASEL
                          000000D0 001D 12485          .LONG   ^X000000D0                ;MOVL
```

```
000000D1  0021 12486          .LONG   ^X000000D1                      ;CMPL
000000D2  0025 12487          .LONG   ^X000000D2                      ;MCOML
000000D3  0029 12488          .LONG   ^X000000D3                      ;BITL
000000D4  002D 12489          .LONG   ^X000000D4                      ;CLRL
000000D5  0031 12490          .LONG   ^X000000D5                      ;TSTL
000000D6  0035 12491          .LONG   ^X000000D6                      ;INCL
000000D7  0039 12492          .LONG   ^X000000D7                      ;DECL
000000D8  003D 12493          .LONG   ^X000000D8                      ;ADWC
000000D9  0041 12494          .LONG   ^X000000D9                      ;SBWC
000000DA  0045 12495          .LONG   ^X000000DA                      ;MTPR
000000DB  0049 12496          .LONG   ^X000000DB                      ;MFPR
000000DC  004D 12497          .LONG   ^X000000DC                      ;MOVPSL
000000DD  0051 12498          .LONG   ^X000000DD                      ;PUSHL
000000DE  0055 12499          .LONG   ^X000000DE                      ;MOVAL
000000DF  0059 12500          .LONG   ^X000000DF                      ;PUSHAL
000000E0  005D 12501          .LONG   ^X000000E0                      ;BBS
000000E1  0061 12502          .LONG   ^X000000E1                      ;BBC
00000000  0065 12503          .LONG   ^X00000000
          0069 12504
          0069 12505 END_CPU_DATA
          0244 12506
          0244 12507
          0244 12508 END_TEST_DATA
          0244
          0244        ;-----------------------------------------------------------------------
          0244 12509 ENDTEST
```

```
                        0022        .SBTTL  "TEST   08D     NATIVE MODE DSIZE ROM TEST 11
                        0022 12511 ;*******************************************************************************
                        0022 12512 ;++
                        0022 12513 ;
                        0022 12514 ; FUNCTIONAL DESCRIPTION:
                        0022 12515 ;
                        0022 12516 ;       THIS IS ONE IN A SERIES OF 14 TESTS THAT CHECK THE CONTENTS OF THE
                        0022 12517 ;       DSIZE ROM.
                        0022 12518 ;
                        0022 12519 ;
                        0022 12520 ; TEST ALGORITHM:
                        0022 12521 ;
                        0022 12522 ;       1.  CREATE AN INDEX (K) TO INDEX RESULTS TABLE
                        0022 12523 ;       2.  CREATE LOOP (I) OF 25
                        0022 12524 ;       3.  SAVE LOOP COUNT IN 1$
                        0022 12525 ;       4.  MOVE LOOP COUNT TO WDREG
                        0022 12526 ;       5.  EXECUTE DCS PROGRAM
                        0022 12527 ;               a.  DISABLE ^P INTERRUPTS
                        0022 12528 ;               b.  SET IPL TO 1F
                        0022 12529 ;               c.  MOVE LOOP COUNT TO RTEMP2
                        0022 12530 ;               d.  ROTATE LOOP COUNT LEFT 2 TO FORM CACHE ADDRESS
                        0022 12531 ;               e.  LOAD THIS ADDRESS INTO PC TO CAUSE PREFETCH
                        0022 12532 ;               f.  PERFORM IRD1TEST TO CAUSE IR TO LOAD INSTRUCTION
                        0022 12533 ;               g.  SET DTYPE TO INSTURCTION DEPENDENT AND STROBE THE VBUS
                        0022 12534 ;       6.  TEST VBUS FOR CORRECT DSIZE
                        0022 12535 ;       7.  IF NO ERROR CREATE LOOP (J) OF 5
                        0022 12536 ;       8.  EXECUTE DCS PROGRAM
                        0022 12537 ;               a.  INCREMENT THE IRD COUNTER
                        0022 12538 ;               b.  SET DTYPE TO INSTRUCTION DEPENDENT AND STROBE THE VBUS
                        0022 12539 ;       9.  TEST VBUS FOR CORRECT DSIZE
                        0022 12540 ;      10.  INCREMENT INDEX K BY ONE
                        0022 12541 ;      11.  IF NO ERROR GO TO STEP 8 UNTIL LOOP J IS EXHAUSTED
                        0022 12542 ;      12.  GO TO STEP 3 UNTIL LOOP I IS EXHAUSTED
                        0022 12543 ;
                        0022 12544 ;
                        0022 12545 ; TEST PATTERNS:
                        0022 12546 ;
                        0022 12547 ;       LOOP COUNT I             INSTRUCTION
                        0022 12548 ;            1                      BBSS
                        0022 12549 ;            2                      BBCS
                        0022 12550 ;            3                      BBSC
                        0022 12551 ;            4                      BBCC
                        0022 12552 ;            5                      BBSSI
                        0022 12553 ;            6                      BBCCI
                        0022 12554 ;            7                      BLBS
                        0022 12555 ;            8                      BLBC
                        0022 12556 ;            9                      FFS
                        0022 12557 ;            A                      FFC
                        0022 12558 ;            B                      CMPV
                        0022 12559 ;            C                      CMPZV
                        0022 12560 ;            D                      EXTV
                        0022 12561 ;            E                      EXTZV
                        0022 12562 ;            F                      INSV
                        0022 12563 ;           10                      ACBL
                        0022 12564 ;           11                      AOBLSS
```

```
                        0022 12565 ;           12                      AOBLEQ
                        0022 12566 ;           13                      SOBGEQ
                        0022 12567 ;           14                      SOBGTR
                        0022 12568 ;           15                      CVTLB
                        0022 12569 ;           16                      CVTLW
                        0022 12570 ;           17                      ASHP
                        0022 12571 ;           18                      CVTLP
                        0022 12572 ;           19                      CALLG
                        0022 12573 ;
                        0022 12574 ;
                        0022 12575 ; LOGIC DESCRIPTION:
                        0022 12576 ;
                        0022 12577 ;       THIS TEST ALONG WITH THE OTHER 13 TESTS IN THIS SERIES CHECK ALL THE
                        0022 12578 ;       LOCATIONS BLASTED INTO THE DSIZE ROM.  ANY FALIURES SHOULD BE CONFINED
                        0022 12579 ;       TO THE ROM.
                        0022 12580 ;
                        0022 12581 ;
                        0022 12582 ; ASSUMPTIONS:
                        0022 12583 ;
                        0022 12584 ;       THIS TEST ASSUMES THE PREFETCH LOGIC AND INSTRUCTION REGISTER ARE
                        0022 12585 ;       OPERATIONAL.
                        0022 12586 ;
                        0022 12587 ;
                        0022 12588 ; ERROR DESCRIPTION:
                        0022 12589 ;
                        0022 12590 ;       EXPECTED VBUS DATA (DSIZE)
                        0022 12591 ;       RECEIVED VBUS DATA
                        0022 12592 ;       LOOP COUNT I
                        0022 12593 ;       LOOP COUNT J
                        0022 12594 ;       LOOP COUNT K
                        0022 12595 ;              (BITS <7:0> = VBUS BIT POSITION)
                        0022 12596 ;              (BITS <8> = BIT STATE)
                        0022 12597 ;
                        0022 12598 ;--
                        0022 12599 ;******************************************************************************
                        0022 12600
                        0022 12601
                        0022 12602          INITIALIZE                              ;INIT THE CPU
                        0024 12603          LDINDEX         K,1                     ;INIT INDEX K TO 1
                        002B 12604          LOOP            I,1,25                  ;CREATE A TEST LOOP OF 50
                        0032 12605          SAVEINDEX       I,1$                    ;SAVE LOOP COUNT IN 1$
                        0037 12606          LOADREG         WDREG,1$,,L             ;LOAD LOOP COUNT INTO WDREG
                        003F 12607          ERRLOOP                                 ;ERROR LOOP POINT
                        0041 12608          FETCH           0                       ;START DCS PROGRAM
                        0046 12609          BRSTCLK         <^X0A>,INH              ;END DCS PROGRAM
                        004A 12610          CMPVBUS         2$,I                    ;TEST DSIZE
                        004F 12611          IFERROR                                 ;ROM FAILURE
                        0054 12612          LOOP            J,1,5                   ;CREATE TEST LOOP OF 5
                        005B 12613          FETCH           <^X0B>                  ;START DCS PROGRAM
                        0060 12614          BRSTCLK         <^X0E>,INH              ;END DCS PROGRAM
                        0064 12615          CMPVBUS         3$,K                    ;TEST DSIZE
                        0069 12616          IFERROR                                 ;ROM FAILURE
                        006E 12617          INCINDEX        K                       ;INCREMENT INDEX K BY ONE
                        0071 12618          ENDLOOP         J                       ;END TEST LOOP J
                        0074 12619          ENDLOOP         I                       ;END TEST LOOP I
```

```
                    0077 12620          SKIP                                    ;GO TO NEXT TEST
                    007E 12621
                    007E 12622
                    007E 12623 BEGIN_TEST_DATA
                    007E
                    007E        ;-----------------------------------------------------------------------
                    007E 12624
00000000            007E 12625 1$:      .LONG   ^X00000000                      ;LOOP COUNT STORAGE
       02           0082 12626 2$:      .BYTE   ^X2                             ;RESULTS TABLE (IRD 0)
                    0083 12627          VBUSDATA        <^X24>,0                ;BBSS
                    0084 12628          VBUSDATA        <^X25>,1
       02           0085 12629          .BYTE   ^X2                             ;BBCS
                    0086 12630          VBUSDATA        <^X24>,0
                    0087 12631          VBUSDATA        <^X25>,1
       02           0088 12632          .BYTE   ^X2                             ;BBSC
                    0089 12633          VBUSDATA        <^X24>,0
                    008A 12634          VBUSDATA        <^X25>,1
       02           008B 12635          .BYTE   ^X2                             ;BBCC
                    008C 12636          VBUSDATA        <^X24>,0
                    008D 12637          VBUSDATA        <^X25>,1
       02           008E 12638          .BYTE   ^X2                             ;BBSSI
                    008F 12639          VBUSDATA        <^X24>,0
                    0090 12640          VBUSDATA        <^X25>,1
       02           0091 12641          .BYTE   ^X2                             ;BBCCI
                    0092 12642          VBUSDATA        <^X24>,0
                    0093 12643          VBUSDATA        <^X25>,1
       02           0094 12644          .BYTE   ^X2                             ;BLBS
                    0095 12645          VBUSDATA        <^X24>,0
                    0096 12646          VBUSDATA        <^X25>,1
       02           0097 12647          .BYTE   ^X2                             ;BLBC
                    0098 12648          VBUSDATA        <^X24>,0
                    0099 12649          VBUSDATA        <^X25>,1
       02           009A 12650          .BYTE   ^X2                             ;FFS
                    009B 12651          VBUSDATA        <^X24>,0
                    009C 12652          VBUSDATA        <^X25>,1
       02           009D 12653          .BYTE   ^X2                             ;FFC
                    009E 12654          VBUSDATA        <^X24>,0
                    009F 12655          VBUSDATA        <^X25>,1
       02           00A0 12656          .BYTE   ^X2                             ;CMPV
                    00A1 12657          VBUSDATA        <^X24>,0
                    00A2 12658          VBUSDATA        <^X25>,1
       02           00A3 12659          .BYTE   ^X2                             ;CMPZV
                    00A4 12660          VBUSDATA        <^X24>,0
                    00A5 12661          VBUSDATA        <^X25>,1
       02           00A6 12662          .BYTE   ^X2                             ;EXTV
                    00A7 12663          VBUSDATA        <^X24>,0
                    00A8 12664          VBUSDATA        <^X25>,1
       02           00A9 12665          .BYTE   ^X2                             ;EXTZV
                    00AA 12666          VBUSDATA        <^X24>,0
                    00AB 12667          VBUSDATA        <^X25>,1
       02           00AC 12668          .BYTE   ^X2                             ;INSV
                    00AD 12669          VBUSDATA        <^X24>,0
                    00AE 12670          VBUSDATA        <^X25>,1
       02           00AF 12671          .BYTE   ^X2                             ;ACBL
                    00B0 12672          VBUSDATA        <^X24>,0
```

```
                00B1 12673          VBUSDATA        <^X25>,1
02              00B2 12674          .BYTE   ^X2                             ;AOBLSS
                00B3 12675          VBUSDATA        <^X24>,0
                00B4 12676          VBUSDATA        <^X25>,1
02              00B5 12677          .BYTE   ^X2                             ;AOBLEQ
                00B6 12678          VBUSDATA        <^X24>,0
                00B7 12679          VBUSDATA        <^X25>,1
02              00B8 12680          .BYTE   ^X2                             ;SOBGEQ
                00B9 12681          VBUSDATA        <^X24>,0
                00BA 12682          VBUSDATA        <^X25>,1
02              00BB 12683          .BYTE   ^X2                             ;SOBGTR
                00BC 12684          VBUSDATA        <^X24>,0
                00BD 12685          VBUSDATA        <^X25>,1
02              00BE 12686          .BYTE   ^X2                             ;CVTLB
                00BF 12687          VBUSDATA        <^X24>,0
                00C0 12688          VBUSDATA        <^X25>,1
02              00C1 12689          .BYTE   ^X2                             ;CVTLW
                00C2 12690          VBUSDATA        <^X24>,0
                00C3 12691          VBUSDATA        <^X25>,1
02              00C4 12692          .BYTE   ^X2                             ;ASHP
                00C5 12693          VBUSDATA        <^X24>,0
                00C6 12694          VBUSDATA        <^X25>,0
02              00C7 12695          .BYTE   ^X2                             ;CVTLP
                00C8 12696          VBUSDATA        <^X24>,0
                00C9 12697          VBUSDATA        <^X25>,1
02              00CA 12698          .BYTE   ^X2                             ;CALLG
                00CB 12699          VBUSDATA        <^X24>,0
                00CC 12700          VBUSDATA        <^X25>,0
                00CD 12701
02              00CD 12702 3$:      .BYTE   ^X2                             ;RESULTS TABLE (IRD 1 TO 5)
                00CE 12703          VBUSDATA        <^X24>,0                ;BBSS
                00CF 12704          VBUSDATA        <^X25>,0
02              00D0 12705          .BYTE   ^X2
                00D1 12706          VBUSDATA        <^X24>,0
                00D2 12707          VBUSDATA        <^X25>,0
02              00D3 12708          .BYTE   ^X2
                00D4 12709          VBUSDATA        <^X24>,0
                00D5 12710          VBUSDATA        <^X25>,0
02              00D6 12711          .BYTE   ^X2
                00D7 12712          VBUSDATA        <^X24>,0
                00D8 12713          VBUSDATA        <^X25>,0
02              00D9 12714          .BYTE   ^X2
                00DA 12715          VBUSDATA        <^X24>,0
                00DB 12716          VBUSDATA        <^X25>,0
02              00DC 12717          .BYTE   ^X2                             ;BBCS
                00DD 12718          VBUSDATA        <^X24>,0
                00DE 12719          VBUSDATA        <^X25>,0
02              00DF 12720          .BYTE   ^X2
                00E0 12721          VBUSDATA        <^X24>,0
                00E1 12722          VBUSDATA        <^X25>,0
02              00E2 12723          .BYTE   ^X2
                00E3 12724          VBUSDATA        <^X24>,0
                00E4 12725          VBUSDATA        <^X25>,0
02              00E5 12726          .BYTE   ^X2
                00E6 12727          VBUSDATA        <^X24>,0
```

```
                00E7 12728          VBUSDATA        <^X25>,0
             02 00E8 12729          .BYTE   ^X2
                00E9 12730          VBUSDATA        <^X24>,0
                00EA 12731          VBUSDATA        <^X25>,0
             02 00EB 12732          .BYTE   ^X2                             ;BBSC
                00EC 12733          VBUSDATA        <^X24>,0
                00ED 12734          VBUSDATA        <^X25>,0
             02 00EE 12735          .BYTE   ^X2
                00EF 12736          VBUSDATA        <^X24>,0
                00F0 12737          VBUSDATA        <^X25>,0
             02 00F1 12738          .BYTE   ^X2
                00F2 12739          VBUSDATA        <^X24>,0
                00F3 12740          VBUSDATA        <^X25>,0
             02 00F4 12741          .BYTE   ^X2
                00F5 12742          VBUSDATA        <^X24>,0
                00F6 12743          VBUSDATA        <^X25>,0
             02 00F7 12744          .BYTE   ^X2
                00F8 12745          VBUSDATA        <^X24>,0
                00F9 12746          VBUSDATA        <^X25>,0
             02 00FA 12747          .BYTE   ^X2                             ;BBCC
                00FB 12748          VBUSDATA        <^X24>,0
                00FC 12749          VBUSDATA        <^X25>,0
             02 00FD 12750          .BYTE   ^X2
                00FE 12751          VBUSDATA        <^X24>,0
                00FF 12752          VBUSDATA        <^X25>,0
             02 0100 12753          .BYTE   ^X2
                0101 12754          VBUSDATA        <^X24>,0
                0102 12755          VBUSDATA        <^X25>,0
             02 0103 12756          .BYTE   ^X2
                0104 12757          VBUSDATA        <^X24>,0
                0105 12758          VBUSDATA        <^X25>,0
             02 0106 12759          .BYTE   ^X2
                0107 12760          VBUSDATA        <^X24>,0
                0108 12761          VBUSDATA        <^X25>,0
             02 0109 12762          .BYTE   ^X2                             ;BBSSI
                010A 12763          VBUSDATA        <^X24>,0
                010B 12764          VBUSDATA        <^X25>,0
             02 010C 12765          .BYTE   ^X2
                010D 12766          VBUSDATA        <^X24>,0
                010E 12767          VBUSDATA        <^X25>,0
             02 010F 12768          .BYTE   ^X2
                0110 12769          VBUSDATA        <^X24>,0
                0111 12770          VBUSDATA        <^X25>,0
             02 0112 12771          .BYTE   ^X2
                0113 12772          VBUSDATA        <^X24>,0
                0114 12773          VBUSDATA        <^X25>,0
             02 0115 12774          .BYTE   ^X2
                0116 12775          VBUSDATA        <^X24>,0
                0117 12776          VBUSDATA        <^X25>,0
             02 0118 12777          .BYTE   ^X2                             ;BBCCI
                0119 12778          VBUSDATA        <^X24>,0
                011A 12779          VBUSDATA        <^X25>,0
             02 011B 12780          .BYTE   ^X2
                011C 12781          VBUSDATA        <^X24>,0
                011D 12782          VBUSDATA        <^X25>,0
```

```
                    02  011E 12783          .BYTE   ^X2
                        011F 12784          VBUSDATA        <^X24>,0
                        0120 12785          VBUSDATA        <^X25>,0
                    02  0121 12786          .BYTE   ^X2
                        0122 12787          VBUSDATA        <^X24>,0
                        0123 12788          VBUSDATA        <^X25>,0
                    02  0124 12789          .BYTE   ^X2
                        0125 12790          VBUSDATA        <^X24>,0
                        0126 12791          VBUSDATA        <^X25>,0
                    02  0127 12792          .BYTE   ^X2                             ;BLBS
                        0128 12793          VBUSDATA        <^X24>,0
                        0129 12794          VBUSDATA        <^X25>,0
                    02  012A 12795          .BYTE   ^X2
                        012B 12796          VBUSDATA        <^X24>,0
                        012C 12797          VBUSDATA        <^X25>,0
                    02  012D 12798          .BYTE   ^X2
                        012E 12799          VBUSDATA        <^X24>,0
                        012F 12800          VBUSDATA        <^X25>,0
                    02  0130 12801          .BYTE   ^X2
                        0131 12802          VBUSDATA        <^X24>,0
                        0132 12803          VBUSDATA        <^X25>,0
                    02  0133 12804          .BYTE   ^X2
                        0134 12805          VBUSDATA        <^X24>,0
                        0135 12806          VBUSDATA        <^X25>,0
                    02  0136 12807          .BYTE   ^X2                             ;BLBC
                        0137 12808          VBUSDATA        <^X24>,0
                        0138 12809          VBUSDATA        <^X25>,0
                    02  0139 12810          .BYTE   ^X2
                        013A 12811          VBUSDATA        <^X24>,0
                        013B 12812          VBUSDATA        <^X25>,0
                    02  013C 12813          .BYTE   ^X2
                        013D 12814          VBUSDATA        <^X24>,0
                        013E 12815          VBUSDATA        <^X25>,0
                    02  013F 12816          .BYTE   ^X2
                        0140 12817          VBUSDATA        <^X24>,0
                        0141 12818          VBUSDATA        <^X25>,0
                    02  0142 12819          .BYTE   ^X2
                        0143 12820          VBUSDATA        <^X24>,0
                        0144 12821          VBUSDATA        <^X25>,0
                    02  0145 12822          .BYTE   ^X2                                 ;FFS
                        0146 12823          VBUSDATA        <^X24>,0
                        0147 12824          VBUSDATA        <^X25>,0
                    02  0148 12825          .BYTE   ^X2
                        0149 12826          VBUSDATA        <^X24>,0
                        014A 12827          VBUSDATA        <^X25>,0
                    02  014B 12828          .BYTE   ^X2
                        014C 12829          VBUSDATA        <^X24>,0
                        014D 12830          VBUSDATA        <^X25>,1
                    02  014E 12831          .BYTE   ^X2
                        014F 12832          VBUSDATA        <^X24>,0
                        0150 12833          VBUSDATA        <^X25>,0
                    02  0151 12834          .BYTE   ^X2
                        0152 12835          VBUSDATA        <^X24>,0
                        0153 12836          VBUSDATA        <^X25>,0
                    02  0154 12837          .BYTE   ^X2                             ;FFC
```

```
                0155 12838          VBUSDATA        <^X24>,0
                0156 12839          VBUSDATA        <^X25>,0
            02  0157 12840          .BYTE   ^X2
                0158 12841          VBUSDATA        <^X24>,0
                0159 12842          VBUSDATA        <^X25>,0
            02  015A 12843          .BYTE   ^X2
                015B 12844          VBUSDATA        <^X24>,0
                015C 12845          VBUSDATA        <^X25>,1
            02  015D 12846          .BYTE   ^X2
                015E 12847          VBUSDATA        <^X24>,0
                015F 12848          VBUSDATA        <^X25>,0
            02  0160 12849          .BYTE   ^X2
                0161 12850          VBUSDATA        <^X24>,0
                0162 12851          VBUSDATA        <^X25>,0
            02  0163 12852          .BYTE   ^X2                             ;CMPV
                0164 12853          VBUSDATA        <^X24>,0
                0165 12854          VBUSDATA        <^X25>,0
            02  0166 12855          .BYTE   ^X2
                0167 12856          VBUSDATA        <^X24>,0
                0168 12857          VBUSDATA        <^X25>,0
            02  0169 12858          .BYTE   ^X2
                016A 12859          VBUSDATA        <^X24>,0
                016B 12860          VBUSDATA        <^X25>,1
            02  016C 12861          .BYTE   ^X2
                016D 12862          VBUSDATA        <^X24>,0
                016E 12863          VBUSDATA        <^X25>,0
            02  016F 12864          .BYTE   ^X2
                0170 12865          VBUSDATA        <^X24>,0
                0171 12866          VBUSDATA        <^X25>,0
            02  0172 12867          .BYTE   ^X2                             ;CMPZV
                0173 12868          VBUSDATA        <^X24>,0
                0174 12869          VBUSDATA        <^X25>,0
            02  0175 12870          .BYTE   ^X2
                0176 12871          VBUSDATA        <^X24>,0
                0177 12872          VBUSDATA        <^X25>,0
            02  0178 12873          .BYTE   ^X2
                0179 12874          VBUSDATA        <^X24>,0
                017A 12875          VBUSDATA        <^X25>,1
            02  017B 12876          .BYTE   ^X2
                017C 12877          VBUSDATA        <^X24>,0
                017D 12878          VBUSDATA        <^X25>,0
            02  017E 12879          .BYTE   ^X2
                017F 12880          VBUSDATA        <^X24>,0
                0180 12881          VBUSDATA        <^X25>,0
            02  0181 12882          .BYTE   ^X2                             ;EXTV
                0182 12883          VBUSDATA        <^X24>,0
                0183 12884          VBUSDATA        <^X25>,0
            02  0184 12885          .BYTE   ^X2
                0185 12886          VBUSDATA        <^X24>,0
                0186 12887          VBUSDATA        <^X25>,0
            02  0187 12888          .BYTE   ^X2
                0188 12889          VBUSDATA        <^X24>,0
                0189 12890          VBUSDATA        <^X25>,1
            02  018A 12891          .BYTE   ^X2
                018B 12892          VBUSDATA        <^X24>,0
```

```
                    018C 12893          VBUSDATA        <^X25>,0
                02  018D 12894          .BYTE   ^X2
                    018E 12895          VBUSDATA        <^X24>,0
                    018F 12896          VBUSDATA        <^X25>,0
                02  0190 12897          .BYTE   ^X2                             ;EXTZV
                    0191 12898          VBUSDATA        <^X24>,0
                    0192 12899          VBUSDATA        <^X25>,0
                02  0193 12900          .BYTE   ^X2
                    0194 12901          VBUSDATA        <^X24>,0
                    0195 12902          VBUSDATA        <^X25>,0
                02  0196 12903          .BYTE   ^X2
                    0197 12904          VBUSDATA        <^X24>,0
                    0198 12905          VBUSDATA        <^X25>,1
                02  0199 12906          .BYTE   ^X2
                    019A 12907          VBUSDATA        <^X24>,0
                    019B 12908          VBUSDATA        <^X25>,0
                02  019C 12909          .BYTE   ^X2
                    019D 12910          VBUSDATA        <^X24>,0
                    019E 12911          VBUSDATA        <^X25>,0
                02  019F 12912          .BYTE   ^X2                             ;INSV
                    01A0 12913          VBUSDATA        <^X24>,0
                    01A1 12914          VBUSDATA        <^X25>,1
                02  01A2 12915          .BYTE   ^X2
                    01A3 12916          VBUSDATA        <^X24>,0
                    01A4 12917          VBUSDATA        <^X25>,0
                02  01A5 12918          .BYTE   ^X2
                    01A6 12919          VBUSDATA        <^X24>,0
                    01A7 12920          VBUSDATA        <^X25>,0
                02  01A8 12921          .BYTE   ^X2
                    01A9 12922          VBUSDATA        <^X24>,0
                    01AA 12923          VBUSDATA        <^X25>,0
                02  01AB 12924          .BYTE   ^X2
                    01AC 12925          VBUSDATA        <^X24>,0
                    01AD 12926          VBUSDATA        <^X25>,0
                02  01AE 12927          .BYTE   ^X2                             ;ACBL
                    01AF 12928          VBUSDATA        <^X24>,0
                    01B0 12929          VBUSDATA        <^X25>,1
                02  01B1 12930          .BYTE   ^X2
                    01B2 12931          VBUSDATA        <^X24>,0
                    01B3 12932          VBUSDATA        <^X25>,1
                02  01B4 12933          .BYTE   ^X2
                    01B5 12934          VBUSDATA        <^X24>,0
                    01B6 12935          VBUSDATA        <^X25>,0
                02  01B7 12936          .BYTE   ^X2
                    01B8 12937          VBUSDATA        <^X24>,0
                    01B9 12938          VBUSDATA        <^X25>,0
                02  01BA 12939          .BYTE   ^X2
                    01BB 12940          VBUSDATA        <^X24>,0
                    01BC 12941          VBUSDATA        <^X25>,0
                02  01BD 12942          .BYTE   ^X2                             ;AOBLSS
                    01BE 12943          VBUSDATA        <^X24>,0
                    01BF 12944          VBUSDATA        <^X25>,1
                02  01C0 12945          .BYTE   ^X2
                    01C1 12946          VBUSDATA        <^X24>,0
                    01C2 12947          VBUSDATA        <^X25>,0
```

```
                02  01C3 12948          .BYTE   ^X2
                    01C4 12949          VBUSDATA        <^X24>,0
                    01C5 12950          VBUSDATA        <^X25>,0
                02  01C6 12951          .BYTE   ^X2
                    01C7 12952          VBUSDATA        <^X24>,0
                    01C8 12953          VBUSDATA        <^X25>,0
                02  01C9 12954          .BYTE   ^X2
                    01CA 12955          VBUSDATA        <^X24>,0
                    01CB 12956          VBUSDATA        <^X25>,0
                02  01CC 12957          .BYTE   ^X2                             ;AOBLEQ
                    01CD 12958          VBUSDATA        <^X24>,0
                    01CE 12959          VBUSDATA        <^X25>,1
                02  01CF 12960          .BYTE   ^X2
                    01D0 12961          VBUSDATA        <^X24>,0
                    01D1 12962          VBUSDATA        <^X25>,0
                02  01D2 12963          .BYTE   ^X2
                    01D3 12964          VBUSDATA        <^X24>,0
                    01D4 12965          VBUSDATA        <^X25>,0
                02  01D5 12966          .BYTE   ^X2
                    01D6 12967          VBUSDATA        <^X24>,0
                    01D7 12968          VBUSDATA        <^X25>,0
                02  01D8 12969          .BYTE   ^X2
                    01D9 12970          VBUSDATA        <^X24>,0
                    01DA 12971          VBUSDATA        <^X25>,0
                02  01DB 12972          .BYTE   ^X2                             ;SOBGEQ
                    01DC 12973          VBUSDATA        <^X24>,0
                    01DD 12974          VBUSDATA        <^X25>,0
                02  01DE 12975          .BYTE   ^X2
                    01DF 12976          VBUSDATA        <^X24>,0
                    01E0 12977          VBUSDATA        <^X25>,0
                02  01E1 12978          .BYTE   ^X2
                    01E2 12979          VBUSDATA        <^X24>,0
                    01E3 12980          VBUSDATA        <^X25>,0
                02  01E4 12981          .BYTE   ^X2
                    01E5 12982          VBUSDATA        <^X24>,0
                    01E6 12983          VBUSDATA        <^X25>,0
                02  01E7 12984          .BYTE   ^X2
                    01E8 12985          VBUSDATA        <^X24>,0
                    01E9 12986          VBUSDATA        <^X25>,0
                02  01EA 12987          .BYTE   ^X2                             ;SOBGTR
                    01EB 12988          VBUSDATA        <^X24>,0
                    01EC 12989          VBUSDATA        <^X25>,0
                02  01ED 12990          .BYTE   ^X2
                    01EE 12991          VBUSDATA        <^X24>,0
                    01EF 12992          VBUSDATA        <^X25>,0
                02  01F0 12993          .BYTE   ^X2
                    01F1 12994          VBUSDATA        <^X24>,0
                    01F2 12995          VBUSDATA        <^X25>,0
                02  01F3 12996          .BYTE   ^X2
                    01F4 12997          VBUSDATA        <^X24>,0
                    01F5 12998          VBUSDATA        <^X25>,0
                02  01F6 12999          .BYTE   ^X2
                    01F7 13000          VBUSDATA        <^X24>,0
                    01F8 13001          VBUSDATA        <^X25>,0
                02  01F9 13002          .BYTE   ^X2                             ;CVTLB
```

```
                01FA 13003          VBUSDATA        <^X24>,0
                01FB 13004          VBUSDATA        <^X25>,0
            02  01FC 13005          .BYTE   ^X2
                01FD 13006          VBUSDATA        <^X24>,0
                01FE 13007          VBUSDATA        <^X25>,0
            02  01FF 13008          .BYTE   ^X2
                0200 13009          VBUSDATA        <^X24>,0
                0201 13010          VBUSDATA        <^X25>,0
            02  0202 13011          .BYTE   ^X2
                0203 13012          VBUSDATA        <^X24>,0
                0204 13013          VBUSDATA        <^X25>,0
            02  0205 13014          .BYTE   ^X2
                0206 13015          VBUSDATA        <^X24>,0
                0207 13016          VBUSDATA        <^X25>,0
            02  0208 13017          .BYTE   ^X2                             ;CVTLW
                0209 13018          VBUSDATA        <^X24>,1
                020A 13019          VBUSDATA        <^X25>,0
            02  020B 13020          .BYTE   ^X2
                020C 13021          VBUSDATA        <^X24>,0
                020D 13022          VBUSDATA        <^X25>,0
            02  020E 13023          .BYTE   ^X2
                020F 13024          VBUSDATA        <^X24>,0
                0210 13025          VBUSDATA        <^X25>,0
            02  0211 13026          .BYTE   ^X2
                0212 13027          VBUSDATA        <^X24>,0
                0213 13028          VBUSDATA        <^X25>,0
            02  0214 13029          .BYTE   ^X2
                0215 13030          VBUSDATA        <^X24>,0
                0216 13031          VBUSDATA        <^X25>,0
            02  0217 13032          .BYTE   ^X2                         ;ASHP
                0218 13033          VBUSDATA        <^X24>,1
                0219 13034          VBUSDATA        <^X25>,0
            02  021A 13035          .BYTE   ^X2
                021B 13036          VBUSDATA        <^X24>,0
                021C 13037          VBUSDATA        <^X25>,0
            02  021D 13038          .BYTE   ^X2
                021E 13039          VBUSDATA        <^X24>,0
                021F 13040          VBUSDATA        <^X25>,0
            02  0220 13041          .BYTE   ^X2
                0221 13042          VBUSDATA        <^X24>,1
                0222 13043          VBUSDATA        <^X25>,0
            02  0223 13044          .BYTE   ^X2
                0224 13045          VBUSDATA        <^X24>,0
                0225 13046          VBUSDATA        <^X25>,0
            02  0226 13047          .BYTE   ^X2                         ;CVTLP
                0227 13048          VBUSDATA        <^X24>,1
                0228 13049          VBUSDATA        <^X25>,0
            02  0229 13050          .BYTE   ^X2
                022A 13051          VBUSDATA        <^X24>,0
                022B 13052          VBUSDATA        <^X25>,0
            02  022C 13053          .BYTE   ^X2
                022D 13054          VBUSDATA        <^X24>,0
                022E 13055          VBUSDATA        <^X25>,0
            02  022F 13056          .BYTE   ^X2
                0230 13057          VBUSDATA        <^X24>,0
```

```
                                  0231 13058          VBUSDATA        <^X25>,0
                              02  0232 13059          .BYTE   ^X2
                                  0233 13060          VBUSDATA        <^X24>,0
                                  0234 13061          VBUSDATA        <^X25>,0
                              02  0235 13062          .BYTE   ^X2                                  ;CALLG
                                  0236 13063          VBUSDATA        <^X24>,0
                                  0237 13064          VBUSDATA        <^X25>,0
                              02  0238 13065          .BYTE   ^X2
                                  0239 13066          VBUSDATA        <^X24>,0
                                  023A 13067          VBUSDATA        <^X25>,0
                              02  023B 13068          .BYTE   ^X2
                                  023C 13069          VBUSDATA        <^X24>,0
                                  023D 13070          VBUSDATA        <^X25>,0
                              02  023E 13071          .BYTE   ^X2
                                  023F 13072          VBUSDATA        <^X24>,0
                                  0240 13073          VBUSDATA        <^X25>,0
                              02  0241 13074          .BYTE   ^X2
                                  0242 13075          VBUSDATA        <^X24>,0
                                  0243 13076          VBUSDATA        <^X25>,0
                                  0244 13077
                                  0244 13078
                                  0244 13079 BEGIN_CPU_DATA
                                  0002 13080
                                  0002 13081 DCS_DATA
                                  0001 13082
U 00, 7700,C800,0364,0300,0470,1801  0001 13083 ;% 00   NOP
U 01, 7700,4800,5BE4,0300,2470,1802  000C 13087 ;% 01   WCTRL/LOADCRAR,WB_R[TEMP0]                ;DISABLE ^P INTERRUPT
U 02, 7700,0800,5BE4,0304,2C70,1803  0017 13091 ;% 02   WCTRL/CONWRITE,WB_R[TEMP1]                ;...
U 03, 7700,9800,D370,030F,FA70,1804  0022 13095 ;% 03   IPL_[1F]                                   ;SET IPL TO 1F
U 04, 5700,4840,0364,0308,0470,1805  002D 13099 ;% 04   R[TEMP2]_RDM                               ;MOVE LOOP COUNT TO RTEMP2
U 05, 7700,D800,EF64,0301,0470,1806  0038 13103 ;% 05   PL_[2]                                     ;SET PLATCH TO 2
U 06, 7700,C800,8B71,0308,0470,1807  0043 13107 ;% 06   Q_R[TEMP2].RL.P                            ;ROTATE LOOP COUNT LEFT 2
U 07, 7700,4800,03A4,03D8,4870,1808  004E 13111 ;% 07   WCTRL/PC_WB,WB_Q                           ;FORCE PREFETCH TO HAPPEN
U 08, 7700,4800,0364,1700,0470,1809  0059 13115 ;% 08   IRD1TEST                                   ;LOAD IR REGISTER
U 09, 7600,4800,0364,0300,0470,180A  0064 13119 ;% 09   DTYPE/IDEP,STROBE.V.BUS                    ;LOAD DSIZE INTO VBUS
U 0A, 7300,C800,0364,0300,0470,180B  006F 13123 ;% 0A   NOP,STOP.CLOCK                             ;STOP THE CPU CLOCK
U 0B, 7700,4800,0364,0300,0470,180C  007A 13127 ;% 0B   NOP
U 0C, 7700,C800,0364,1B00,0470,180D  0085 13131 ;% 0C   LOD INC BRA?                               ;INCREMENT THE IRD COUNTER
U 0D, 7600,C800,0364,0300,0470,180E  0090 13135 ;% 0D   DTYPE/IDEP,STROBE.V.BUS                    ;LOAD DSIZE INTO VBUS
U 0E, 7300,4800,0364,0300,0470,180F  009B 13139 ;% 0E   NOP,STOP.CLOCK                             ;STOP THE CPU CLOCK
                                  00A6 13143
                                  00A6 13144 RTEMP_DATA
                                  0001 13145
                          00C00000  0001 13146          .LONG   ^X00C00000                         ;DISABLE ^P INTERRUPT
                          00400000  0005 13147          .LONG   ^X00400000                         ;...
                                  0009 13148
                                  0009 13149 CACHE_DATA
                                  0001 13150
                          000000E2  0001 13151          .LONG   ^X000000E2                         ;BBSS
                          000000E3  0005 13152          .LONG   ^X000000E3                         ;BBCS
                          000000E4  0009 13153          .LONG   ^X000000E4                         ;BBSC
                          000000E5  000D 13154          .LONG   ^X000000E5                         ;BBCC
                          000000E6  0011 13155          .LONG   ^X000000E6                         ;BBSSI
                          000000E7  0015 13156          .LONG   ^X000000E7                         ;BBCCI
                          000000E8  0019 13157          .LONG   ^X000000E8                         ;BLBS
```

```
000000E9  001D 13158          .LONG   ^X000000E9                      ;BLBC
000000EA  0021 13159          .LONG   ^X000000EA                      ;FFS
000000EB  0025 13160          .LONG   ^X000000EB                      ;FFC
000000EC  0029 13161          .LONG   ^X000000EC                      ;CMPV
000000ED  002D 13162          .LONG   ^X000000ED                      ;CMPZV
000000EE  0031 13163          .LONG   ^X000000EE                      ;EXTV
000000EF  0035 13164          .LONG   ^X000000EF                      ;EXTZV
000000F0  0039 13165          .LONG   ^X000000F0                      ;INSV
000000F1  003D 13166          .LONG   ^X000000F1                      ;ACBL
000000F2  0041 13167          .LONG   ^X000000F2                      ;AOBLSS
000000F3  0045 13168          .LONG   ^X000000F3                      ;AOBLEQ
000000F4  0049 13169          .LONG   ^X000000F4                      ;SOBGEQ
000000F5  004D 13170          .LONG   ^X000000F5                      ;SOBGTR
000000F6  0051 13171          .LONG   ^X000000F6                      ;CVTLB
000000F7  0055 13172          .LONG   ^X000000F7                      ;CVTLW
000000F8  0059 13173          .LONG   ^X000000F8                      ;ASHP
000000F9  005D 13174          .LONG   ^X000000F9                      ;CVTLP
000000FA  0061 13175          .LONG   ^X000000FA                      ;CALLG
00000000  0065 13176          .LONG   ^X00000000
          0069 13177
          0069 13178 END_CPU_DATA
          0244 13179
          0244 13180
          0244 13181 END_TEST_DATA
          0244
          0244        ;-----------------------------------------------------------------------
          0244 13182 ENDTEST
```

```
                        0022        .SBTTL  "TEST   08E     NATIVE MODE DSIZE ROM TEST 12
                        0022 13184 ;*******************************************************************************
                        0022 13185 ;++
                        0022 13186 ;
                        0022 13187 ; FUNCTIONAL DESCRIPTION:
                        0022 13188 ;
                        0022 13189 ;       THIS IS ONE IN A SERIES OF 14 TESTS THAT CHECK THE CONTENTS OF THE
                        0022 13190 ;       DSIZE ROM.
                        0022 13191 ;
                        0022 13192 ;
                        0022 13193 ; TEST ALGORITHM:
                        0022 13194 ;
                        0022 13195 ;       1.  CREATE AN INDEX (K) TO INDEX RESULTS TABLE
                        0022 13196 ;       2.  CREATE LOOP (I) OF 5
                        0022 13197 ;       3.  SAVE LOOP COUNT IN 1$
                        0022 13198 ;       4.  MOVE LOOP COUNT TO WDREG
                        0022 13199 ;       5.  EXECUTE DCS PROGRAM
                        0022 13200 ;               a.  DISABLE ^P INTERRUPTS
                        0022 13201 ;               b.  SET IPL TO 1F
                        0022 13202 ;               c.  MOVE LOOP COUNT TO RTEMP2
                        0022 13203 ;               d.  ROTATE LOOP COUNT LEFT 2 TO FORM CACHE ADDRESS
                        0022 13204 ;               e.  LOAD THIS ADDRESS INTO PC TO CAUSE PREFETCH
                        0022 13205 ;               f.  PERFORM IRD1TEST TO CAUSE IR TO LOAD INSTRUCTION
                        0022 13206 ;               g.  SET DTYPE TO INSTURCTION DEPENDENT AND STROBE THE VBUS
                        0022 13207 ;       6.  TEST VBUS FOR CORRECT DSIZE
                        0022 13208 ;       7.  IF NO ERROR CREATE LOOP (J) OF 5
                        0022 13209 ;       8.  EXECUTE DCS PROGRAM
                        0022 13210 ;               a.  INCREMENT THE IRD COUNTER
                        0022 13211 ;               b.  SET DTYPE TO INSTRUCTION DEPENDENT AND STROBE THE VBUS
                        0022 13212 ;       9.  TEST VBUS FOR CORRECT DSIZE
                        0022 13213 ;      10.  INCREMENT INDEX K BY ONE
                        0022 13214 ;      11.  IF NO ERROR GO TO STEP 8 UNTIL LOOP J IS EXHAUSTED
                        0022 13215 ;      12.  GO TO STEP 3 UNTIL LOOP I IS EXHAUSTED
                        0022 13216 ;
                        0022 13217 ;
                        0022 13218 ; TEST PATTERNS:
                        0022 13219 ;
                        0022 13220 ;       LOOP COUNT I             INSTRUCTION
                        0022 13221 ;            1                      CALLS
                        0022 13222 ;            2                      XFC
                        0022 13223 ;            3                      ESCD
                        0022 13224 ;            4                      EXCE
                        0022 13225 ;            5                      ESCF
                        0022 13226 ;
                        0022 13227 ;
                        0022 13228 ; LOGIC DESCRIPTION:
                        0022 13229 ;
                        0022 13230 ;       THIS TEST ALONG WITH THE OTHER 13 TESTS IN THIS SERIES CHECK ALL THE
                        0022 13231 ;       LOCATIONS BLASTED INTO THE DSIZE ROM.  ANY FALIURES SHOULD BE CONFINED
                        0022 13232 ;       TO THE ROM.
                        0022 13233 ;
                        0022 13234 ;
                        0022 13235 ; ASSUMPTIONS:
                        0022 13236 ;
                        0022 13237 ;       THIS TEST ASSUMES THE PREFETCH LOGIC AND INSTRUCTION REGISTER ARE
```

```
                     0022 13238 ;       OPERATIONAL.
                     0022 13239 ;
                     0022 13240 ;
                     0022 13241 ; ERROR DESCRIPTION:
                     0022 13242 ;
                     0022 13243 ;       EXPECTED VBUS DATA (DSIZE)
                     0022 13244 ;       RECEIVED VBUS DATA
                     0022 13245 ;       LOOP COUNT I
                     0022 13246 ;       LOOP COUNT J
                     0022 13247 ;       LOOP COUNT K
                     0022 13248 ;              (BITS <7:0> = VBUS BIT POSITION)
                     0022 13249 ;              (BITS <8> = BIT STATE)
                     0022 13250 ;
                     0022 13251 ;--
                     0022 13252 ;*******************************************************************************
                     0022 13253
                     0022 13254
                     0022 13255         INITIALIZE                              ;INIT THE CPU
                     0024 13256         LDINDEX         K,1                     ;INIT INDEX K TO 1
                     002B 13257         LOOP            I,1,5                   ;CREATE A TEST LOOP OF 50
                     0032 13258         SAVEINDEX       I,1$                    ;SAVE LOOP COUNT IN 1$
                     0037 13259         LOADREG         WDREG,1$,,L             ;LOAD LOOP COUNT INTO WDREG
                     003F 13260         ERRLOOP                                 ;ERROR LOOP POINT
                     0041 13261         FETCH           0                       ;START DCS PROGRAM
                     0046 13262         BRSTCLK         <^X0A>,INH              ;END DCS PROGRAM
                     004A 13263         CMPVBUS         2$,I                    ;TEST DSIZE
                     004F 13264         IFERROR                                 ;ROM FAILURE
                     0054 13265         LOOP            J,1,5                   ;CREATE TEST LOOP OF 5
                     005B 13266         FETCH           <^X0B>                  ;START DCS PROGRAM
                     0060 13267         BRSTCLK         <^X0E>,INH              ;END DCS PROGRAM
                     0064 13268         CMPVBUS         3$,K                    ;TEST DSIZE
                     0069 13269         IFERROR                                 ;ROM FAILURE
                     006E 13270         INCINDEX        K                       ;INCREMENT INDEX K BY ONE
                     0071 13271         ENDLOOP         J                       ;END TEST LOOP J
                     0074 13272         ENDLOOP         I                       ;END TEST LOOP I
                     0077 13273         SKIP                                    ;GO TO NEXT TEST
                     007E 13274
                     007E 13275
                     007E 13276 BEGIN_TEST_DATA
                     007E
                     007E       ;-------------------------------------------------------------------------
                     007E 13277
00000000             007E 13278 1$:     .LONG   ^X00000000                      ;LOOP COUNT STORAGE
       02            0082 13279 2$:     .BYTE   ^X2                             ;RESULTS TABLE (IRD 0)
                     0083 13280         VBUSDATA        <^X24>,0                ;CALLS
                     0084 13281         VBUSDATA        <^X25>,1
       02            0085 13282         .BYTE   ^X2                             ;XFC
                     0086 13283         VBUSDATA        <^X24>,0
                     0087 13284         VBUSDATA        <^X25>,1
       02            0088 13285         .BYTE   ^X2                             ;ESCD
                     0089 13286         VBUSDATA        <^X24>,1
                     008A 13287         VBUSDATA        <^X25>,1
       02            008B 13288         .BYTE   ^X2                             ;ESCE
                     008C 13289         VBUSDATA        <^X24>,0
                     008D 13290         VBUSDATA        <^X25>,0
```

```
                    02  008E 13291          .BYTE   ^X2                             ;ESCF
                        008F 13292          VBUSDATA        <^X24>,0
                        0090 13293          VBUSDATA        <^X25>,0
                        0091 13294
                    02  0091 13295 3$:      .BYTE   ^X2                             ;RESULTS TABLE (IRD 1 TO 5)
                        0092 13296          VBUSDATA        <^X24>,0                ;CALLS
                        0093 13297          VBUSDATA        <^X25>,0
                    02  0094 13298          .BYTE   ^X2
                        0095 13299          VBUSDATA        <^X24>,0
                        0096 13300          VBUSDATA        <^X25>,0
                    02  0097 13301          .BYTE   ^X2
                        0098 13302          VBUSDATA        <^X24>,0
                        0099 13303          VBUSDATA        <^X25>,0
                    02  009A 13304          .BYTE   ^X2
                        009B 13305          VBUSDATA        <^X24>,0
                        009C 13306          VBUSDATA        <^X25>,0
                    02  009D 13307          .BYTE   ^X2
                        009E 13308          VBUSDATA        <^X24>,0
                        009F 13309          VBUSDATA        <^X25>,0
                    02  00A0 13310          .BYTE   ^X2                             ;XFC
                        00A1 13311          VBUSDATA        <^X24>,0
                        00A2 13312          VBUSDATA        <^X25>,1
                    02  00A3 13313          .BYTE   ^X2
                        00A4 13314          VBUSDATA        <^X24>,0
                        00A5 13315          VBUSDATA        <^X25>,1
                    02  00A6 13316          .BYTE   ^X2
                        00A7 13317          VBUSDATA        <^X24>,0
                        00A8 13318          VBUSDATA        <^X25>,1
                    02  00A9 13319          .BYTE   ^X2
                        00AA 13320          VBUSDATA        <^X24>,0
                        00AB 13321          VBUSDATA        <^X25>,1
                    02  00AC 13322          .BYTE   ^X2
                        00AD 13323          VBUSDATA        <^X24>,0
                        00AE 13324          VBUSDATA        <^X25>,1
                    02  00AF 13325          .BYTE   ^X2                             ;ESCD
                        00B0 13326          VBUSDATA        <^X24>,1
                        00B1 13327          VBUSDATA        <^X25>,1
                    02  00B2 13328          .BYTE   ^X2
                        00B3 13329          VBUSDATA        <^X24>,1
                        00B4 13330          VBUSDATA        <^X25>,1
                    02  00B5 13331          .BYTE   ^X2
                        00B6 13332          VBUSDATA        <^X24>,1
                        00B7 13333          VBUSDATA        <^X25>,1
                    02  00B8 13334          .BYTE   ^X2
                        00B9 13335          VBUSDATA        <^X24>,1
                        00BA 13336          VBUSDATA        <^X25>,1
                    02  00BB 13337          .BYTE   ^X2
                        00BC 13338          VBUSDATA        <^X24>,1
                        00BD 13339          VBUSDATA        <^X25>,1
                    02  00BE 13340          .BYTE   ^X2                             ;ESCE
                        00BF 13341          VBUSDATA        <^X24>,0
                        00C0 13342          VBUSDATA        <^X25>,0
                    02  00C1 13343          .BYTE   ^X2
                        00C2 13344          VBUSDATA        <^X24>,0
                        00C3 13345          VBUSDATA        <^X25>,0
```

```
                    02  00C4 13346          .BYTE   ^X2
                        00C5 13347          VBUSDATA        <^X24>,0
                        00C6 13348          VBUSDATA        <^X25>,0
                    02  00C7 13349          .BYTE   ^X2
                        00C8 13350          VBUSDATA        <^X24>,0
                        00C9 13351          VBUSDATA        <^X25>,0
                    02  00CA 13352          .BYTE   ^X2
                        00CB 13353          VBUSDATA        <^X24>,0
                        00CC 13354          VBUSDATA        <^X25>,0
                    02  00CD 13355          .BYTE   ^X2                             ;ESCF
                        00CE 13356          VBUSDATA        <^X24>,0
                        00CF 13357          VBUSDATA        <^X25>,0
                    02  00D0 13358          .BYTE   ^X2
                        00D1 13359          VBUSDATA        <^X24>,0
                        00D2 13360          VBUSDATA        <^X25>,0
                    02  00D3 13361          .BYTE   ^X2
                        00D4 13362          VBUSDATA        <^X24>,0
                        00D5 13363          VBUSDATA        <^X25>,0
                    02  00D6 13364          .BYTE   ^X2
                        00D7 13365          VBUSDATA        <^X24>,0
                        00D8 13366          VBUSDATA        <^X25>,0
                    02  00D9 13367          .BYTE   ^X2
                        00DA 13368          VBUSDATA        <^X24>,0
                        00DB 13369          VBUSDATA        <^X25>,0
                        00DC 13370
                        00DC 13371
                        00DC 13372 BEGIN_CPU_DATA
                        0002 13373
                        0002 13374 DCS_DATA
                        0001 13375
U 00, 7700,C800,0364,0300,0470,1801     0001 13376 ;% 00    NOP
U 01, 7700,4800,5BE4,0300,2470,1802     000C 13380 ;% 01    WCTRL/LOADCRAR,WB_R[TEMP0]      ;DISABLE ^P INTERRUPT
U 02, 7700,0800,5BE4,0304,2C70,1803     0017 13384 ;% 02    WCTRL/CONWRITE,WB_R[TEMP1]      ;...
U 03, 7700,9800,D370,030F,FA70,1804     0022 13388 ;% 03    IPL_[1F]                        ;SET IPL TO 1F
U 04, 5700,4840,0364,0308,0470,1805     002D 13392 ;% 04    R[TEMP2]_RDM                    ;MOVE LOOP COUNT TO RTEMP2
U 05, 7700,D800,EF64,0301,0470,1806     0038 13396 ;% 05    PL_[2]                          ;SET PLATCH TO 2
U 06, 7700,C800,8B71,0308,0470,1807     0043 13400 ;% 06    Q_R[TEMP2].RL.P                 ;ROTATE LOOP COUNT LEFT 2
U 07, 7700,4800,03A4,03D8,4870,1808     004E 13404 ;% 07    WCTRL/PC_WB,WB_Q                ;FORCE PREFETCH TO HAPPEN
U 08, 7700,4800,0364,1700,0470,1809     0059 13408 ;% 08    IRD1TEST                        ;LOAD IR REGISTER
U 09, 7600,4800,0364,0300,0470,180A     0064 13412 ;% 09    DTYPE/IDEP,STROBE.V.BUS         ;LOAD DSIZE INTO VBUS
U 0A, 7300,C800,0364,0300,0470,180B     006F 13416 ;% 0A    NOP,STOP.CLOCK                  ;STOP THE CPU CLOCK
U 0B, 7700,4800,0364,0300,0470,180C     007A 13420 ;% 0B    NOP
U 0C, 7700,C800,0364,1B00,0470,180D     0085 13424 ;% 0C    LOD INC BRA?                    ;INCREMENT THE IRD COUNTER
U 0D, 7600,C800,0364,0300,0470,180E     0090 13428 ;% 0D    DTYPE/IDEP,STROBE.V.BUS         ;LOAD DSIZE INTO VBUS
U 0E, 7300,4800,0364,0300,0470,180F     009B 13432 ;% 0E    NOP,STOP.CLOCK                  ;STOP THE CPU CLOCK
                        00A6 13436
                        00A6 13437 RTEMP_DATA
                        0001 13438
                    00C00000    0001 13439          .LONG   ^X00C00000              ;DISABLE ^P INTERRUPT
                    00400000    0005 13440          .LONG   ^X00400000              ;...
                        0009 13441
                        0009 13442 CACHE_DATA
                        0001 13443
                    000000FB    0001 13444          .LONG   ^X000000FB              ;CALLS
                    000000FC    0005 13445          .LONG   ^X000000FC              ;XFC
```

```
000000FD  0009 13446          .LONG   ^X000000FD                      ;ESCD
000000FE  000D 13447          .LONG   ^X000000FE                      ;ESCE
000000FF  0011 13448          .LONG   ^X000000FF                      ;ESCF
00000000  0015 13449          .LONG   ^X00000000
          0019 13450
          0019 13451 END_CPU_DATA
          00DC 13452
          00DC 13453
          00DC 13454 END_TEST_DATA
          00DC
          00DC          ;-----------------------------------------------------------------------
          00DC 13455 ENDTEST
```

```
                0000 13457 .END
```

```
$CONTROL_C_CODE= 0000000A
$FILE_NOT_FOUND= 00000009
$STN            = 00000000
$TEST_NUMBER    = 0000008F
ACV             = 0000000F
AC_LOW          = 00000080
ADD             = 00000019
ADDR_MATCH_HI   = 00007405
ADDR_MATCH_LO   = 00007404
ADK             = 0000000B
ALK             = 00000000
ALP             = 00000016
ARG_CNT         = 0000006A R      ED
A_REG_BYTE_0    = 00007410
A_REG_BYTE_1    = 00007411
A_REG_BYTE_2    = 00007412
A_REG_BYTE_3    = 00007413
B               = 00000001
BADD1           = 00007426
BELL_FLAG       = 00000008
BELL_KEY        = 0000000E
BUFF_ADDR_LOW   = 00007426
BURST_STOP_FLAG= 00000080
BYTE            = 00000001
CACHE_FLAG      = 00000000
CACHE_FLAG1     = 00000000
CACHE_HEAD      = 00000000 R      F0
CACHE_LENGTH    = 00000001
CAK             = 0000000A
CCC             = 00000004
CCS             = 00000005
CF_KEY          = 00000024
CLA             = 00000003
CLEAR_CTRL_FILE= 00000002
CLKDCS          = 00007427
CLK_CTRL_0      = 00000020
CLK_CTRL_0_RO   = 00000040
CLK_CTRL_1      = 00000040
CLK_CTRL_1_RO   = 00000080
CLOCK_STOPED    = 00000080
CMI             = 00000007
CMICTL          = 0000741C
CMI_COMPLETE    = 00000020
CMI_CTRL_CLEAR  = 00000001
CMI_CTRL_REG    = 0000741C
CMI_GO          = 00000008
CMI_READ        = 00000040
CMI_STATUS_0    = 00000008
CMI_STATUS_1    = 00000010
CMI_WRITE       = 00000020
CML             = 0000000E
COMPOSIT_LENGTH= 00000004
CON             = 00000011
CONTINUE_FLAG   = 00000002
CONTROL_C_FLAG  = 00000001
CONT_FLAG       = 00000020
CONT_KEY        = 0000001C
CPU_RUN         = 00000010
CSAMR           = 00007404
CSBUF           = 00007424
CSTRP           = 00007422
CS_ADD_BUFF_HGH= 00007425
CS_ADD_BUFF_LOW= 00007424
CS_ADD_TRAP_HGH= 00007423
CS_ADD_TRAP_LOW= 00007422
CYCLE_FLAG      = 00000008
CYCLE_KEY       = 00000020
D0              = 0000000E
D1              = 00000008
D2              = 00000000
DATA_FLAG       = 00000000
DATA_LENGTH     = 00000002
DCSAD           = 00007401
DCSCF           = 00007403
DCSCR           = 00007402
DCSDA           = 00007400
DCS_ADDR_CLEAR  = 00000002
DCS_ADDR_REG_RO= 00007427
DCS_ADDR_REG_WO= 00007401
DCS_CTRL_FILE   = 00007403
DCS_CTRL_REG    = 00007402
DCS_DATA_REG    = 00007400
DCS_FLAG        = 00000000
DCS_FLAG1       = 00000000
DCS_HEAD        = 00000000 R      EE
DCS_LENGTH      = 00000001
DCS_SCRATCH_ADR= 00000036
DCS_START       = 00001800
DC_LOW          = 00000040
DIAGNOSE_FLAG   = 00000004
DPM             = 00000000
DUM1            = FFFFFFFE
ENDTEST_LOC     = 000000E1 R      ED
END_SLICE       = 00000025
END_TEST_100      000000FD R      05
END_TEST_101      00000175 R      07
END_TEST_102      00000058 R      09
END_TEST_103      000000F4 R      0F
END_TEST_104      0000034B R      11
END_TEST_105      00000105 R      17
END_TEST_106      0000014A R      19
END_TEST_107      00000434 R      1F
END_TEST_108      00000228 R      21
END_TEST_109      0000007C R      27
END_TEST_110      000000D8 R      2D
END_TEST_111      00000079 R      33
END_TEST_112      0000009F R      39
END_TEST_113      000001A8 R      3F
END_TEST_114      00000044 R      45
END_TEST_115      0000005D R      4B
END_TEST_116      0000005C R      51
END_TEST_117      00000070 R      57
END_TEST_118      00000071 R      5D
END_TEST_119      000000DF R      63
END_TEST_120      00000078 R      69
END_TEST_121      00000078 R      6F
END_TEST_122      000000DF R      75
END_TEST_123      00000081 R      7B
END_TEST_124      0000006B R      81
END_TEST_125      00000178 R      87
END_TEST_126      0000016C R      8D
END_TEST_127      000000B8 R      93
END_TEST_128      000000B2 R      99
END_TEST_129      000000C9 R      9F
END_TEST_130      00000057 R      A5
END_TEST_131      00000098 R      AB
END_TEST_132      00000243 R      B1
END_TEST_133      00000243 R      B7
END_TEST_134      00000243 R      BD
END_TEST_135      00000243 R      C3
END_TEST_136      00000243 R      C9
END_TEST_137      00000243 R      CF
END_TEST_138      00000243 R      D5
END_TEST_139      00000243 R      DB
END_TEST_140      00000244 R      E1
END_TEST_141      00000244 R      E7
END_TEST_142      000000DC R      ED
END_TEST_99       0000013A R      02
EN_TRACE_REG    = 00000008
ERRLOG_FLAG     = 00000000
ERROR_FLAG      = 00000010
ERROR_LOG_ADR   = 000032FC
ERR_LOG_INIT    = 00000001
EXP_STALL_FLAG = 00000010
EXP_UTRAP_FLAG = 00000040
FAC             = 00000020
FAULT           = 00000002
FCC             = 00000015
FCS             = 00000021
FETCH_FLAG      = 00000008
FEX             = 0000001E
FFH             = 00000024
FFL             = 00000023
FIC             = 0000001F
FIO             = 0000001D
FLAG_KEY        = 00000004
FMR             = 00000022
FPA             = 00000002
FP_BOOT_0       = 00000001
FP_BOOT_1       = 00000002
FP_START_0      = 00000004
FP_START_1      = 00000008
FQA             = 00000014
FRONT1          = 00007420
FRONT2          = 00007421
```

```
FRONT_PNL_1     = 00007420          MC1             = 00000009          OP_SKIP         = 0000001C
FRONT_PNL_2     = 00007421          MDL             = 0000001B          OP_SUB_TEST     = 0000001E
FRONT_PNL_LOCK  = 00000080          MDR             = 00000018          OVERLAY_HEAD    = 00000000 R     F2
HALT_FLAG       = 00000001          MDREG           = 00007414          PARITY_ERROR    = 00000020
HALT_KEY        = 00000008          MD_KEY          = 00000034          PAR_CHK_ENABL   = 00000008
HALT_ON_MATCH   = 00000001          MD_REG_BYTE_0   = 00007414          PAR_STOP_ENABL  = 00000010
HEAD            = FFFF7C00          MD_REG_BYTE_1   = 00007415          PASS_KEY        = 00000002
HIGH            = 00000001          MD_REG_BYTE_2   = 00007416          PB_INIT         = 00000040
H_FROM_WBUS     = 00000010          MD_REG_BYTE_3   = 00007417          PB_KEY          = 00000036
IB_FLAG         = 00000020          MEC             = 0000001A          PC_KEY          = 00000030
IB_KEY          = 00000012          MHREG           = 00007418          PHB             = 00000007
INSTR_FLAG      = 00000004          MH_REG_BYTE_0   = 00007418          PRK             = 0000000C
INSTR_KEY       = 00000018          MH_REG_BYTE_1   = 00007419          PS_KEY          = 0000003E
INT             = 00000010          MH_REG_BYTE_2   = 0000741A          QA_FLAG         = 00000040
IRD             = 00000005          MH_REG_BYTE_3   = 0000741B          QA_KEY          = 00000014
I_INDEX_FLAG    = 00000000          MIC             = 00000001          QV_FLAG         = 00000002
I_INIT          = 00000000          MICROWORD       = 0000000A          QV_KEY          = 00000028
J_INDEX_FLAG    = 00000000          MICRO_ADDR_INH  = 00000080          RDCTRL          = 0000741E
J_INIT          = 00000000          MONITOR_SIZE    = 00002C5E          RDM             = 00000004
K_INDEX_FLAG    = 00000000          MSQ             = 00000008          RDM_FLAG        = 00000004
K_INIT          = 00000000          MT_KEY          = 0000002E          RDM_KEY         = 0000002A
L               = 00000004          N               = 00000001          RDM_LAST        = 00000008
LIST_END        = 0000006E R     ED NER_FLAG        = 00000004          RD_CTRL_REG     = 0000741E
LIST_HEAD       = 0000006B R     ED NER_KEY         = 0000000C          REMOTE          = 00000001
LIST_LENGTH     = 00000003          NOERROR         = 00000001          RTEMP_FLAG      = 00000000
LITERAL         = 00000000          NOTEQUAL        = 00000003          RTEMP_FLAG1     = 00000000
LONG            = 00000004          ONEQUAL         = 00000002          RTEMP_HEAD      = 00000000 R     EF
LOOP_ERROR_FLAG= 00000020           ONERROR         = 00000000          RTEMP_LENGTH    = 00000001
LOOP_FLAG       = 00000002          OP_BEGIN_LOOP   = 00000008          RT_KEY          = 0000002C
LOOP_KEY        = 0000000A          OP_BURST_CLOCK  = 00000002          SAC             = 00000009
LOST_FLAG       = 00000010          OP_COMPARE_REG  = 00000004          SA_CLOCK        = 00000080
LOST_KEY        = 0000001A          OP_COMPARE_REGM= 00000006           SA_FLAG         = 00000010
LOW             = 00000000          OP_COMPARE_VB   = 00000020          SA_KEY          = 00000010
M               = 0000000A          OP_DUMPLOG      = 00000030          SA_START_STOP   = 00000080
M$TEST_SIZE     = 000007A2          OP_ENABL_STALL  = 00000038          SBE_FLAG        = 00000020
M$TEST_SIZE_D   = 000003E2          OP_ENABL_TRAP   = 0000002E          SBE_KEY         = 00000042
MA0             = 00000008          OP_END_FILE     = 00000024          SECTION_FLAG    = 00000080
MA1             = 0000000A          OP_END_LOOP     = 0000000A          SF_KEY          = 00000040
MAD             = 00000013          OP_END_TEST     = 0000000C          SOMM_FLAG       = 00000001
MAIN32          = 0000741D          OP_ERRLOG       = 00000032          SOMM_KEY        = 00000006
MAINT_A_TO_CMI  = 00000080          OP_ERROR_LOOP   = 0000000E          SPA             = 00000002
MAINT_CMI_TO_MH= 00000020           OP_FETCH        = 00000022          SRK             = 00000001
MAINT_DCS_ENABL= 00000004           OP_IF_ERROR     = 00000012          SRM             = 00000017
MAINT_MD_TO_CMI= 00000040           OP_INC_INDEX    = 0000002A          SR_KEY          = 0000003C
MAINT_REG       = 0000741D          OP_INIT         = 00000014          STACK_SIZE      = 00000400
MAINT_STB_INH   = 00000002          OP_LOAD_DCS     = 00000000          STATUS          = 00007406
MAINT_TRAP_OFF  = 00000001          OP_LOAD_FIELD   = 00000026          STATUS_REG      = 00007406
MAINT_WB_TO_WH  = 00000008          OP_LOAD_INDEX   = 00000028          STEP_KEY        = 0000001E
MAINT_WD_TO_WB  = 00000010          OP_LOAD_REG     = 00000016          STOP_CLOCK      = 00000004
MAP             = 00000012          OP_MASK         = 00000018          STROBE_CMI      = 00000040
MAREG           = 00007410          OP_NEW_TEST     = 0000001A          ST_KEY          = 0000001E
MASTER_HALT_EN  = 00000008          OP_PATTERN_GEN  = 00000010          TEMP            = 00000002 R     EC
MAX_ARGUMENTS   = 00000010          OP_PRINT_N      = 00000036          TEMP1           = 00000036
MA_KEY          = 00000038          OP_PRINT_S      = 00000034          TEMP2           = 000000C8
MC0             = 00000003          OP_SAVE_INDEX   = 0000002C          TEMP3           = 00000002
```

```
TEMP4           = 00000002 R     EC
TEST            = 00000004
TEST_FLAG       = 00000000
TEST_HEAD       = 0000001F R     ED
TEST_KEY        = 00000000
TEST_LENGTH     = 00000002
TEST_SECT_HEAD  = 00000000 R     ED
THIS_TEST       = 0000008E
TICK_FLAG       = 00000002
TICK_KEY        = 00000022
TOK             = 00000006
TRACE_ENABL     = 00000004
TRAP_HALT_EN    = 00000020
TRAP_OFF        = 00000002
TR_FLAG         = 00000080
TR_KEY          = 00000016
TSTSPAN_FLAG    = 00000040
UBI             = 00000006
UDP             = 0000001C
UTR             = 0000000D
VA_KEY          = 00000032
VBUS_CLOCK      = 00000080
VBUS_COMPARE    = 00000080
VBUS_LOAD       = 00000010
VBUS_SERIAL_IN  = 00000040
VBUS_SERIAL_OUT= 00000001
VB_KEY          = 00000026
V_BUS_STROBE    = 00000001
W               = 00000002
WBUS_FROM_D     = 00000020
WDREG           = 0000742C
WD_KEY          = 0000003A
WD_REG_BYTE_0   = 0000742C
WD_REG_BYTE_1   = 0000742D
WD_REG_BYTE_2   = 0000742E
WD_REG_BYTE_3   = 0000742F
WHREG           = 00007430
WH_REG_BYTE_0   = 00007430
WH_REG_BYTE_1   = 00007431
WH_REG_BYTE_2   = 00007432
WH_REG_BYTE_3   = 00007433
WORD            = 00000002
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
X_0099_D	00000002 (2.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0099_T	0000017E (382.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
DIRECTORY	0000002C (44.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0100_D	00000002 (2.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0100_T	0000017E (382.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0101_D	00000002 (2.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0101_T	0000017E (382.)	07 (7.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0102_D	00000002 (2.)	08 (8.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0102_T	00000080 (128.)	09 (9.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0102_GDCS	0000004E (78.)	0A (10.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0102_ERTEMP	00000001 (1.)	0B (11.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0102_FCACHE	00000001 (1.)	0C (12.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0102_HFILL	0000002E (46.)	0D (13.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0103_D	00000002 (2.)	0E (14.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0103_T	000000FE (254.)	0F (15.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0104_D	00000002 (2.)	10 (16.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0104_T	00000380 (896.)	11 (17.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0104_GDCS	000001C4 (452.)	12 (18.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0104_ERTEMP	00000025 (37.)	13 (19.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0104_FCACHE	00000001 (1.)	14 (20.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0104_HFILL	00000014 (20.)	15 (21.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0105_D	00000002 (2.)	16 (22.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0105_T	0000017E (382.)	17 (23.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0106_D	00000002 (2.)	18 (24.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0106_T	00000180 (384.)	19 (25.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0106_GDCS	000000BC (188.)	1A (26.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0106_ERTEMP	00000015 (21.)	1B (27.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0106_FCACHE	00000001 (1.)	1C (28.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0106_HFILL	0000002C (44.)	1D (29.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0107_D	00000002 (2.)	1E (30.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0107_T	0000047E (1150.)	1F (31.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0108_D	00000002 (2.)	20 (32.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0108_T	00000280 (640.)	21 (33.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0108_GDCS	000000E8 (232.)	22 (34.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0108_ERTEMP	00000021 (33.)	23 (35.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0108_FCACHE	00000001 (1.)	24 (36.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0108_HFILL	00000074 (116.)	25 (37.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0109_D	00000002 (2.)	26 (38.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0109_T	00000100 (256.)	27 (39.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0109_GDCS	0000002D (45.)	28 (40.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0109_ERTEMP	00000001 (1.)	29 (41.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0109_FCACHE	00000001 (1.)	2A (42.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0109_HFILL	0000004F (79.)	2B (43.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0110_D	00000002 (2.)	2C (44.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0110_T	00000100 (256.)	2D (45.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0110_GDCS	00000038 (56.)	2E (46.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0110_ERTEMP	00000011 (17.)	2F (47.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

X_0110_FCACHE	00000001	(1.)	30	(48.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0110_HFILL	00000034	(52.)	31	(49.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0111_D	00000002	(2.)	32	(50.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0111_T	00000080	(128.)	33	(51.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0111_GDCS	0000002D	(45.)	34	(52.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0111_ERTEMP	00000001	(1.)	35	(53.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0111_FCACHE	00000001	(1.)	36	(54.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0111_HFILL	0000004F	(79.)	37	(55.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0112_D	00000002	(2.)	38	(56.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0112_T	00000100	(256.)	39	(57.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0112_GDCS	00000038	(56.)	3A	(58.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0112_ERTEMP	00000009	(9.)	3B	(59.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0112_FCACHE	00000001	(1.)	3C	(60.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0112_HFILL	0000003C	(60.)	3D	(61.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0113_D	00000002	(2.)	3E	(62.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0113_T	00000200	(512.)	3F	(63.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0113_GDCS	00000090	(144.)	40	(64.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0113_ERTEMP	00000011	(17.)	41	(65.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0113_FCACHE	00000001	(1.)	42	(66.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0113_HFILL	0000005C	(92.)	43	(67.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0114_D	00000002	(2.)	44	(68.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0114_T	00000080	(128.)	45	(69.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0114_GDCS	0000002D	(45.)	46	(70.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0114_FCACHE	00000009	(9.)	47	(71.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0114_ERTEMP	00000001	(1.)	48	(72.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0114_HFILL	00000047	(71.)	49	(73.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0115_D	00000002	(2.)	4A	(74.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0115_T	00000080	(128.)	4B	(75.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0115_GDCS	00000059	(89.)	4C	(76.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0115_ERTEMP	0000000D	(13.)	4D	(77.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0115_FCACHE	00000009	(9.)	4E	(78.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0115_HFILL	0000000F	(15.)	4F	(79.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0116_D	00000002	(2.)	50	(80.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0116_T	00000080	(128.)	51	(81.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0116_GDCS	0000004E	(78.)	52	(82.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0116_ERTEMP	00000009	(9.)	53	(83.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0116_FCACHE	00000009	(9.)	54	(84.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0116_HFILL	0000001E	(30.)	55	(85.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0117_D	00000002	(2.)	56	(86.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0117_T	00000080	(128.)	57	(87.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0117_GDCS	0000002D	(45.)	58	(88.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0117_ERTEMP	00000001	(1.)	59	(89.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0117_FCACHE	00000001	(1.)	5A	(90.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0117_HFILL	0000004F	(79.)	5B	(91.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0118_D	00000002	(2.)	5C	(92.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0118_T	00000080	(128.)	5D	(93.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0118_GDCS	00000038	(56.)	5E	(94.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0118_ERTEMP	00000005	(5.)	5F	(95.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0118_FCACHE	00000001	(1.)	60	(96.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0118_HFILL	00000040	(64.)	61	(97.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0119_D	00000002	(2.)	62	(98.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0119_T	00000100	(256.)	63	(99.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0119_GDCS	00000043	(67.)	64	(100.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0119_ERTEMP	00000005	(5.)	65	(101.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0119_FCACHE	00000009	(9.)	66	(102.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE

X_0119_HFILL	0000002D	(45.)	67	(103.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0120_D	00000002	(2.)	68	(104.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0120_T	00000080	(128.)	69	(105.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0120_GDCS	00000064	(100.)	6A	(106.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0120_ERTEMP	0000000D	(13.)	6B	(107.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0120_FCACHE	00000009	(9.)	6C	(108.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0120_HFILL	00000004	(4.)	6D	(109.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0121_D	00000002	(2.)	6E	(110.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0121_T	00000080	(128.)	6F	(111.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0121_GDCS	00000064	(100.)	70	(112.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0121_ERTEMP	0000000D	(13.)	71	(113.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0121_FCACHE	00000009	(9.)	72	(114.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0121_HFILL	00000004	(4.)	73	(115.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0122_D	00000002	(2.)	74	(116.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0122_T	00000100	(256.)	75	(117.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0122_GDCS	000000C7	(199.)	76	(118.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0122_ERTEMP	00000021	(33.)	77	(119.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0122_FCACHE	00000009	(9.)	78	(120.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0122_HFILL	0000000D	(13.)	79	(121.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0123_D	00000002	(2.)	7A	(122.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0123_T	00000100	(256.)	7B	(123.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0123_GDCS	00000090	(144.)	7C	(124.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0123_ERTEMP	00000011	(17.)	7D	(125.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0123_FCACHE	00000009	(9.)	7E	(126.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0123_HFILL	00000054	(84.)	7F	(127.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0124_D	00000002	(2.)	80	(128.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0124_T	00000080	(128.)	81	(129.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0124_GDCS	00000059	(89.)	82	(130.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0124_ERTEMP	00000009	(9.)	83	(131.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0124_FCACHE	00000009	(9.)	84	(132.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0124_HFILL	00000013	(19.)	85	(133.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0125_D	00000002	(2.)	86	(134.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0125_T	00000180	(384.)	87	(135.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0125_GDCS	0000002D	(45.)	88	(136.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0125_ERTEMP	00000001	(1.)	89	(137.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0125_FCACHE	00000001	(1.)	8A	(138.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0125_HFILL	0000004F	(79.)	8B	(139.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0126_D	00000002	(2.)	8C	(140.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0126_T	00000180	(384.)	8D	(141.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0126_GDCS	00000038	(56.)	8E	(142.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0126_ERTEMP	00000001	(1.)	8F	(143.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0126_FCACHE	00000001	(1.)	90	(144.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0126_HFILL	00000044	(68.)	91	(145.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0127_D	00000002	(2.)	92	(146.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0127_T	00000100	(256.)	93	(147.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0127_GDCS	00000038	(56.)	94	(148.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0127_ERTEMP	00000001	(1.)	95	(149.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0127_FCACHE	00000001	(1.)	96	(150.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0127_HFILL	00000044	(68.)	97	(151.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0128_D	00000002	(2.)	98	(152.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0128_T	00000100	(256.)	99	(153.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0128_GDCS	0000002D	(45.)	9A	(154.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0128_ERTEMP	00000001	(1.)	9B	(155.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0128_FCACHE	00000001	(1.)	9C	(156.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0128_HFILL	0000004F	(79.)	9D	(157.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE

X_0129_D	00000002	(2.)	9E	(158.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0129_T	00000100	(256.)	9F	(159.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0129_GDCS	00000038	(56.)	A0	(160.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0129_ERTEMP	00000001	(1.)	A1	(161.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0129_FCACHE	00000001	(1.)	A2	(162.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0129_HFILL	00000044	(68.)	A3	(163.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0130_D	00000002	(2.)	A4	(164.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0130_T	00000080	(128.)	A5	(165.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0130_GDCS	000000A6	(166.)	A6	(166.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0130_ERTEMP	00000001	(1.)	A7	(167.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0130_FCACHE	00000001	(1.)	A8	(168.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0130_HFILL	00000056	(86.)	A9	(169.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0131_D	00000002	(2.)	AA	(170.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0131_T	00000100	(256.)	AB	(171.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0131_GDCS	00000085	(133.)	AC	(172.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0131_ERTEMP	00000009	(9.)	AD	(173.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0131_FCACHE	00000009	(9.)	AE	(174.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0131_HFILL	00000067	(103.)	AF	(175.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0132_D	00000002	(2.)	B0	(176.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0132_T	00000280	(640.)	B1	(177.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0132_GDCS	000000A6	(166.)	B2	(178.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0132_ERTEMP	00000009	(9.)	B3	(179.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0132_FCACHE	00000069	(105.)	B4	(180.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0132_HFILL	00000066	(102.)	B5	(181.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0133_D	00000002	(2.)	B6	(182.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0133_T	00000280	(640.)	B7	(183.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0133_GDCS	000000A6	(166.)	B8	(184.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0133_ERTEMP	00000009	(9.)	B9	(185.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0133_FCACHE	00000069	(105.)	BA	(186.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0133_HFILL	00000066	(102.)	BB	(187.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0134_D	00000002	(2.)	BC	(188.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0134_T	00000280	(640.)	BD	(189.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0134_GDCS	000000A6	(166.)	BE	(190.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0134_ERTEMP	00000009	(9.)	BF	(191.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0134_FCACHE	00000069	(105.)	C0	(192.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0134_HFILL	00000066	(102.)	C1	(193.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0135_D	00000002	(2.)	C2	(194.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0135_T	00000280	(640.)	C3	(195.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0135_GDCS	000000A6	(166.)	C4	(196.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0135_ERTEMP	00000009	(9.)	C5	(197.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0135_FCACHE	00000069	(105.)	C6	(198.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0135_HFILL	00000066	(102.)	C7	(199.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0136_D	00000002	(2.)	C8	(200.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0136_T	00000280	(640.)	C9	(201.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0136_GDCS	000000A6	(166.)	CA	(202.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0136_ERTEMP	00000009	(9.)	CB	(203.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0136_FCACHE	00000069	(105.)	CC	(204.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0136_HFILL	00000066	(102.)	CD	(205.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0137_D	00000002	(2.)	CE	(206.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0137_T	00000280	(640.)	CF	(207.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0137_GDCS	000000A6	(166.)	D0	(208.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0137_ERTEMP	00000009	(9.)	D1	(209.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0137_FCACHE	00000069	(105.)	D2	(210.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0137_HFILL	00000066	(102.)	D3	(211.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0138_D	00000002	(2.)	D4	(212.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE

X_0138_T	00000280	(640.)	D5 (213.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0138_GDCS	000000A6	(166.)	D6 (214.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0138_ERTEMP	00000009	(9.)	D7 (215.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0138_FCACHE	00000069	(105.)	D8 (216.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0138_HFILL	00000066	(102.)	D9 (217.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0139_D	00000002	(2.)	DA (218.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0139_T	00000280	(640.)	DB (219.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0139_GDCS	000000A6	(166.)	DC (220.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0139_ERTEMP	00000009	(9.)	DD (221.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0139_FCACHE	00000069	(105.)	DE (222.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0139_HFILL	00000066	(102.)	DF (223.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0140_D	00000002	(2.)	E0 (224.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0140_T	00000280	(640.)	E1 (225.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0140_GDCS	000000A6	(166.)	E2 (226.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0140_ERTEMP	00000009	(9.)	E3 (227.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0140_FCACHE	00000069	(105.)	E4 (228.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0140_HFILL	00000066	(102.)	E5 (229.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0141_D	00000002	(2.)	E6 (230.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0141_T	00000280	(640.)	E7 (231.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0141_GDCS	000000A6	(166.)	E8 (232.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0141_ERTEMP	00000009	(9.)	E9 (233.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0141_FCACHE	00000069	(105.)	EA (234.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0141_HFILL	00000066	(102.)	EB (235.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0142_D	00000002	(2.)	EC (236.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0142_T	00000100	(256.)	ED (237.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0142_GDCS	000000A6	(166.)	EE (238.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0142_ERTEMP	00000009	(9.)	EF (239.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0142_FCACHE	00000019	(25.)	F0 (240.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0142_HFILL	00000036	(54.)	F1 (241.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0143_D	00000000	(0.)	F2 (242.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	50	00:00:00.18	00:00:00.71
Command processing	875	00:00:00.96	00:00:01.85
Pass 1	4624	00:02:49.80	00:03:39.92
Symbol table sort	0	00:00:00.61	00:00:00.62
Pass 2	43	00:00:49.46	00:01:04.26
Symbol table output	2	00:00:00.22	00:00:00.25
Psect synopsis output	0	00:00:00.81	00:00:01.11
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	5594	00:03:42.04	00:04:48.73

The working set limit was 7800 pages.
1975230 bytes (3858 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 372 non-local and 225 local symbols.
14199 source lines were read in Pass 1, producing 525 object records in Pass 2.
102 pages of virtual memory were used to define 47 macros.

```
                              +---------------------------+
                              ! Macro library statistics !
                              +---------------------------+

Macro library name                          Macros defined
------------------                          --------------
DISK$UCODPACK00:[VAX750]COMETMAC.MLB;805              47
SYS$COMMON:[SYSLIB]STARLET.MLB;2                       0
TOTALS (all libraries)                                47

1960 GETS were required to define 47 macros.

There were 0 errors, 1 warning and 0 information messages, on lines:
 2507 (1)

MACRO UCOD$0:[VAX750.DPM]TITLE+DISK$UCODPACK00:[VAX750]COMETASM+UCOD$0:[VAX750.DPM]ECKAB3/LIS=ECKAB3/OBJ=ECKAB3
```

(1) 24 "DPM Revision History"
(1) 1 ""
(1) 3 "DIAGNOSTIC MICROCODE MACROS"
(1) 98 " RDM Control File Macros"
(1) 109 " Diagnostic Macros"
(1) 194 ""
(1) 195 "MODULE GATE ARRAY MAPS"
(1) 197 " L0001 MULTIPLIER AND GATE ARRAY LOCATION"
(1) 254 " L0002 GATE ARRAY LOCATION"
(1) 311 " L0003 GATE ARRAY LOCATION"
(1) 369 " L0004 GATE ARRAY LOCATION"
(1) 427 " L0011 & L0016 GATE ARRAY LOCATION"
(1) 485 ""
(1) 490 "EQUATED SYMBOLS
(1) 493 ""
(1) 2 "TEST 08F N BIT TEST 1
(1) 299 "TEST 090 N BIT TEST 2
(1) 518 "TEST 091 N BIT TEST 3
(1) 850 "TEST 092 N BIT TEST 4
(1) 1044 "TEST 093 Z BIT TEST 1
(1) 1312 "TEST 094 Z BIT TEST 2
(1) 1525 "TEST 095 V BIT TEST 1
(1) 1792 "TEST 096 V BIT TEST 2
(1) 2047 "TEST 097 C BIT TEST 1
(1) 2329 "TEST 098 C BIT TEST 2
(1) 2557 "TEST 099 BRANCH TEST NATIVE MODE
(1) 2796 "TEST 09A BRANCH TEST COMPATABILITY MODE
(1) 3055 "TEST 09B TEST INTEGER OVERFLOW ARITHMETIC TRAP
(1) 3262 "TEST 09C TEST DECIMAL OVERFLOW ARITHMETIC TRAP
(1) 3474 "TEST 09D ALU STATE FLAGS TEST 1
(1) 3692 "TEST 09E ALU STATE FLAGS TEST 2
(1) 3853 "TEST 09F TEST BUT/CCBR1.CCBR0.IR0
(1) 4016 "TEST 0A0 BUT ON WMUX EQUAL/NOT EQUAL ZERO
(1) 4168 "TEST 0A1 Rotator Test 50 SRK Status Test 1
(1) 4737 "TEST 0A2 Rotator Test 51 SRK Status Test 2
(1) 5135 "TEST 0A3 Rotator Test 52 SRK Status Test 3
(1) 5339 "TEST 0A4 Rotator Test 53 SRK Status Test 4
(1) 5544 "TEST 0A5 Rotator Test 54 SRK Status Test 5
(1) 5810 "TEST 0A6 TEST BUT/CCBR0.SRKSTA0
(1) 5934 "TEST 0A7 TEST BUT/BUTXB
(1) 6141 "TEST 0A8 TEST BUT/PSLTP AND BUT/PSLC
(1) 6276 "TEST 0A9 TEST BUT/REGMODE
(1) 6431 "TEST 0AA TEST BUT/CCBR1.INT-TS
(1) 6653 "TEST 0AB TEST BUT/BCDCHK
(1) 6835 "TEST 0AC TEST BUT/FPS3
(1) 6966 "TEST 0AD TEST BUT/NO.FPA
(1) 7069 "TEST 0AE TEST BUT/FPS2
(1) 7161 "TEST 0AF TEST BUT/FPS1
(1) 7327 "TEST 0B0 BUT/SPASTA TEST 1
(1) 7450 "TEST 0B1 BUT/SPASTA TEST 2
(1) 7566 "TEST 0B2 BUT/SPASTA TEST 3
(1) 7677 "TEST 0B3 BUT/SPASTA TEST 4
(1) 7782 "TEST 0B4 PRINT MICRO CODE REVISION
(1) 7894 "TEST 0B5 PRINT HARDWARE REVISION LEVEL

```
0000     1          .TITLE  ECKAB VAX-11/750 DPM MICRO DIAGNOSTICS
0000     2          .IDENT  /V09.01/
0000     3 ;
0000     4 ; COPYRIGHT (C) 1980,1983
0000     5 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     6 ;
0000     7 ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE  ONLY ON A SINGLE
0000     8 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000     9 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    10 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    11 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    12 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    13 ; REMAIN IN DEC.
0000    14 ;
0000    15 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    16 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    17 ; CORPORATION.
0000    18 ;
0000    19 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    20 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    21 ;
0000    22 ;
```

```
0000     24          .SBTTL  "DPM Revision History"
0000     25 ;++
0000     26 ; DPM MICRO-DIAGNOSTIC PACKAGE
0000     27 ;
0000     28 ; REVISION HISTORY
0000     29 ;
0000     30 ; 00-01 Tom Groetzinger 4-FEB-1980
0000     31 ;       Started file.
0000     32 ;
0000     33 ; 00-02 Dave Spain      27-FEB-1980
0000     34 ;       Added CS BUS INTEGRITY test.
0000     35 ;
0000     36 ; 00-03 Tom Groetzinger 12-MAR-1980
0000     37 ;       Added five tests.
0000     38 ;
0000     39 ; 00-04 Tom Groetzinger 17-MAR-1980
0000     40 ;       Added SRK,SRM callouts to some tests.
0000     41 ;
0000     42 ; 00-05 Tom Groetzinger 17-MAR-1980
0000     43 ;       Improved tests 7 and 16 to increase coverage due to Westboro feedback
0000     44 ;
0000     45 ; 00-06 Tom Groetzinger 17-MAR-1980
0000     46 ;       Debugged test 16 dueu to major rewrite
0000     47 ;
0000     48 ; 00-07 Tom Groetzinger 17-MAR-1980
0000     49 ;       Added 9 Super Rotator tests
0000     50 ;
0000     51 ; 01-08 Dave Spain       4-APR-80
0000     52 ;       Added 4 Micro-sequencer tests for offical release.
0000     53 ;
0000     54 ; 01-09 Bill Landry     4-APR-1980
0000     55 ;       Added the XB, PREFETCH and IRD tests from ECKAC.SRC so
0000     56 ;       manufacturing can run one tape for DPM.
0000     57 ;
0000     58 ; 00-10 Bill Landry     4-APR-1980
0000     59 ;       Let's save version 1.0 for our first SDC release.
0000     60 ;
0000     61 ; 00-11 Tom Groetzinger 7-APR-1980
0000     62 ;       Added 2 ALK Tests
0000     63 ;
0000     64 ; 00-12 Tom Groetzinger 8-APR-1980
0000     65 ;       Deleted last 2 D Register outputs from ALK Test #6
0000     66 ;
0000     67 ; 00-13 Tom Groetzinger 14-APR-1980
0000     68 ;       Added D Register output to ALK test #6
0000     69 ;
0000     70 ; 00-14 Dave Spain      25-APR-1980
0000     71 ;       Changed mask on the micro-sequencer tests to ignore bit 13 of the
0000     72 ;       Control Store address.
0000     73 ;
0000     74 ; 00-15 Dave Spain      28-APR-1980
0000     75 ;       Fixed bug in ALK Tests 6 and 7 by replacing VSIZE/1 micro order with a
0000     76 ;       SIZE[LONG] macro in micro words 6 and 3 respectively.
0000     77 ;
0000     78 ; 00-16 Dave Spain      29-APR-1980
```

```
0000     79 ;       Changed mask on CS address for Instruction Address Decode test from
0000     80 ;       3FFF to 1FFF to ignore bit 13 of Control Store address.
0000     81 ;
0000     82 ; 00-17 Dave Spain      30-APR-1980
0000     83 ;       Added CCS module to error callout for Control Store Bus Integrity test.
0000     84 ;
0000     85 ; 00-18 Dave Spain       9-MAY-1980
0000     86 ;       Added SRM gate array to error callout routine in RTMP0 and RTEMP
0000     87 ;       Sratchpad tests.
0000     88 ;
0000     89 ; 00-19 Bill Landry     10-JUN-1980
0000     90 ;       Added RDM tests to beginning of tape
0000     91 ;
0000     92 ; 01-00 Bill Landry/Dave Spain  13-JUN-1980
0000     93 ;       Added macro file to title section and compatibility mode IRD tests
0000     94 ;       then submitted to RELEASE ENGINERRING.
0000     95 ;
0000     96 ; 01-01 Dave Spain      19-JUN-1980
0000     97 ;       Fixed prefetch incrementer test to clear out cache data after every
0000     98 ;       prefetch. This will prevent the test from mistaking bad addresses from
0000     99 ;       good ones.
0000    100 ;
0000    101 ; 01-02 Dave Spain       2-JUL-1980
0000    102 ;       Added CM Second IRDX tests to file.
0000    103 ;
0000    104 ; 01-03 Dave Spain      18-AUG-1980
0000    105 ;       Added CM branch test to file.
0000    106 ;
0000    107 ; 01-04 Dave Spain      10-SEPT-1980
0000    108 ;       Fixed "DEFAULT MTEMP0 TEST"
0000    109 ;
0000    110 ; 01-05 Bill Landry     11-SEPT-1980
0000    111 ;       Revised RDM to CMI tests
0000    112 ;
0000    113 ; 02-00 Dave Spain      28-SEPT-1980
0000    114 ;       Added rotator tests to file and submitted to SDC.
0000    115 ;
0000    116 ; 02-01 Dave Spain      28-OCT-1980
0000    117 ;       Added QV flag to tests to be run by manufacturing.
0000    118 ;
0000    119 ; 02-02 Bill Landry     10-NOV-1980
0000    120 ;       Added tests for TOK and SAC gate arrays.
0000    121 ;
0000    122 ; 02-03 Bill Landry     13-NOV-1980
0000    123 ;       Added test of REI CHECK
0000    124 ;
0000    125 ; 03-00 Dave Spain      17-NOV-1980
0000    126 ;       Added rotator tests 23-28 and released to SDC.
0000    127 ;
0000    128 ; 03-01 Dave Spain      10-DEC-1980
0000    129 ;       Added rotator tests 29-42.
0000    130 ;
0000    131 ; 03-02 Bill Landry     11-DEC-1980
0000    132 ;       Fixed bug in CS ADDRESS test
0000    133 ;
```

```
0000   134 ; 03-03 Dave Spain      12-DEC-1980
0000   135 ;       Enhanced Rotator Test 1 to call out ALP slices.
0000   136 ;
0000   137 ; 03-04 Bill Landry     16-DEC-1980
0000   138 ;       Added DSIZE ROM tests
0000   139 ;
0000   140 ; 03-05 Bill Landry     31-DEC-1980
0000   141 ;       Last great act of 80. Separated DPM and RDM tests into two files.
0000   142 ;
0000   143 ; 04-00 Bill Landry     5-JAN-1981
0000   144 ;       Submitted to RELEASE ENGINERRING
0000   145 ;
0000   146 ; 04-01 Bill Landry     14-JAN-1981
0000   147 ;       Revised INTERVAL TIMER INTERRUPT TEST to fix bug caused by INT PEND
0000   148 ;       ECO
0000   149 ;
0000   150 ; 04-02 Dave Spain      16-JAN-1981
0000   151 ;       Fixed error callout in CS Parity Error, XB, and Prefetch Incrementer
0000   152 ;       tests and added seven new rotator tests.
0000   153 ;
0000   154 ; 04-03 Bill Landry     4-FEB-1981
0000   155 ;       Added tests for CCC gate array
0000   156 ;
0000   157 ; 04-04 Dave Spain      9-FEB-1981
0000   158 ;       Changed the names of rotator tests 23-33.
0000   159 ;
0000   160 ; 04-05 Bill Landry     24-FEB-1981
0000   161 ;       Added tests for DPM discrete branching logic.
0000   162 ;
0000   163 ; 04-06 Bill Landry     27-FEB-1981
0000   164 ;       Added tests for branching logic in PHB gate array.
0000   165 ;
0000   166 ; 05-00 Dave Spain       1-MAR-1981
0000   167 ;       Added 5 SRK status rotator tests and submitted to SDC.
0000   168 ;
0000   169 ; 05-01 Dave Spain      25-MAR-1981
0000   170 ;       Fixed a loop on test bug in the Alu Functions Test by adding a LDFIELD
0000   171 ;       pseudo-op that reinitializes the ROT field in Subtest 1.
0000   172 ;
0000   173 ; 05-02 Bill Landry     25-MAR-1981
0000   174 ;       Fixed bug in BUT/NO.FPA test.  I was branching on the wrong conditions.
0000   175 ;
0000   176 ; 05-03 Dave Spain      14-APR-1981
0000   177 ;       Cleaned up documentation for R Scratchpads Write All Cells Test.
0000   178 ;
0000   179 ; 06-00 Bill Landry     2-MAY-1981
0000   180 ;       Added SPA branching, CON registers, TU58 registers, and CCS PARITY
0000   181 ;       tests.  Submitted to Release Engineering.
0000   182 ;
0000   183 ; 06-01 Dave Spain      17-JUN-1981
0000   184 ;       Replaced the old IRD1 Instruction Address Decode test with two new
0000   185 ;       tests that toggle the PSL's FPD bit and can handle an FPA.
0000   186 ;
0000   187 ; 06-02 Bill Landry     19-AUG-1981
0000   188 ;       Fixed bugs created by introduction of FPA option.
```

```
0000    189 ;       Replaced all references to COMET with the word CPU.
0000    190 ;       Fixed typing error in vbus data of CPU CLOCK TEST.
0000    191 ;       Submitted to RELEASE ENGINEERING.
0000    192 ;
0000    193 ; 06-03 Bill Landry     24-SEPT-1981
0000    194 ;       Fixed potential clock problem with FPA in Z BIT TEST 1.
0000    195 ;
0000    196 ; 06-04 Bill Landry     9-OCT-1981
0000    197 ;       Added changes for ECO L0002-TW002
0000    198 ;
0000    199 ; 07-00 Bill Landry     31-OCT-1981
0000    200 ;       Removed changes for ECO L0002-TW002.  ECO is on hold.
0000    201 ;       Added tests for FPA to CPU interface logic
0000    202 ;       Moved WCS test to MIC tape
0000    203 ;       Submitted to RELEASE ENGINEERING.
0000    204 ;
0000    205 ; 07-01 Bill Landry     13-NOV-1981
0000    206 ;       Stopped release of 7.0
0000    207 ;       Added FPA TRAP LOGIC TEST
0000    208 ;       Enhanced error messages for FPA
0000    209 ;       Resubmitted to RELEASE ENGINEERING
0000    210 ;
0000    211 ; 07-02 Bill Landry     20-NOV-1981
0000    212 ;       Stopped release of 7.1
0000    213 ;       Enhanced FPA ENABLE/DISABLE TEST to check for MSEQ INIT
0000    214 ;       Resubmitted to RELEASE ENGINEERING
0000    215 ;
0000    216 ; 07-03 Dave Shull      26-Mar-1982
0000    217 ;       Added the ERRLOOP pseudo to the Rotator Extract tests which were
0000    218 ;       missing these and couldn't be looped on.
0000    219 ;
0000    220 ; 07-04 Dave Shull      05-Apr-1982
0000    221 ;       Removed all BEGINSA and ENDSA pseudo's.
0000    222 ;       Submitted to RELEASE ENGINEERING
0000    223 ;
0000    224 ; 07-05 Dave Shull      21-May-1982
0000    225 ;       Add a subtest to the Micro Sequencer Stack Test that will test bits
0000    226 ;       11 and 12 of the micro stack.
0000    227 ;
0000    228 ; 07-06 Dave Shull      24-May-1982
0000    229 ;       Changed documentation in the CM Odd Address BUT Test to reflect the
0000    230 ;       ECO L0002-TW002
0000    231 ;
0000    232 ; 08-00 Dave Shull      18-Jun-1982
0000    233 ;       Added 10 new tests.  Two of those tests are an addition to the existing
0000    234 ;       IRD1 rom tests.  These 2 new tests check the signal ROM OS INH H.
0000    235 ;       The other 8 new tests check the validity of the IRDX ROMs.  They check
0000    236 ;       both the address and the signal ROM OS INH H.
0000    237 ;
0000    238 ; 08-01 Dave Shull      22-Jun-1982
0000    239 ;       Added a subtest to the Register Backup Stack test that will check the
0000    240 ;       bit that records Increment/Decrement of the GPR.
0000    241 ;
0000    242 ; 09-00 Dave Shull      03-Nov-1982
0000    243 ;       Added two new Tests for coverage the new PCS (L0008) module.
```

```
0000   244 ;
0000   245 ; 09-01 Dave Smith      15-May-1983
0000   246 ;       Correct DCS in ALK test 3,6,7. Disabled ALP's on the WBUS.
0000   247 ;--
0000     1          .SBTTL  ""
```

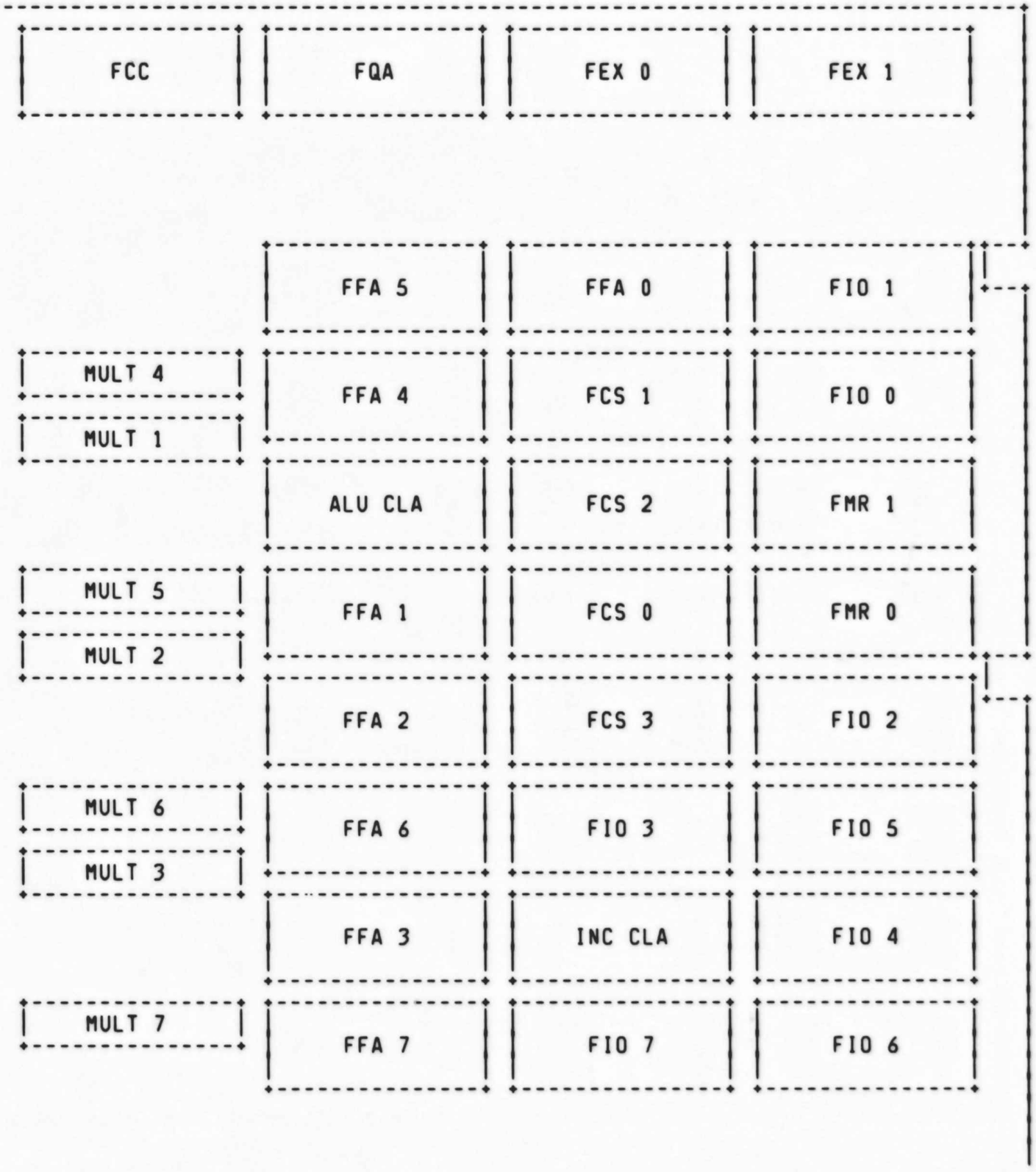
```
0000      3          .SBTTL  "DIAGNOSTIC MICROCODE MACROS"
0000      4 ;++
0000      5 ; Revision History
0000      6 ;
0000      7 ; 28    Dave Spain      22-Jun-82
0000      8 ;       Removed LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
0000      9 ;       Decided not to use them.
0000     10 ;
0000     11 ; 27    Dave Spain      15-Jun-82
0000     12 ;       Added LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
0000     13 ;
0000     14 ; 26    Dave Spain      19-May-82
0000     15 ;       Added M[]_FPA macro.
0000     16 ;
0000     17 ; 25    Rich Lewis      23-Feb-82
0000     18 ;       Added PSL_WB, M[]_R[].OR.NIB0(M), WB_FPA CCR, WB_FPA TCR, Q_R[].ASL.P,
0000     19 ;       RDM_M[].XZ.OR.R[], FPA_WB macros
0000     20 ;
0000     21 ; 24    Dave Spain      02-Feb-82
0000     22 ;       Added FPA_R[]+M[] macro.
0000     23 ;
0000     24 ; 23    Bill Landry     22-OCT-81
0000     25 ;       Added PSL_RDM macro
0000     26 ;
0000     27 ; 22    Dave Spain       5 Oct 81
0000     28 ;       Added RDM_FPA and FPA_M[] FPA_RDM macros.
0000     29 ;
0000     30 ; 21    Bill Landry     29 Sept 81
0000     31 ;       Added FPA macro, FPA_M[] FPA_R[].
0000     32 ;
0000     33 ; 20    Dave Spain      14 Aug 81
0000     34 ;       Added the TESTFPA [] macro.
0000     35 ;
0000     36 ; 19    Dave Spain      18 July 81
0000     37 ;       Added the following FPA macros, FPA_RDM and FPA_R[].
0000     38 ;
0000     39 ; 18    Dave Spain      24 Sept 80
0000     40 ;       Added RDM_Q macro. This macro uses the ALU.
0000     41 ;
0000     42 ; 17    Bill Landry     23 Sept 80
0000     43 ;       Added macro VA_VA-ZLIT0[] for testing memory in decending order.
0000     44 ;
0000     45 ; 16    Dave Spain      27 August 80
0000     46 ;       Changed RDM_Q macro to RDM_Q Q_D. This more accurately reflects macro.
0000     47 ;
0000     48 ; 15    Bill Landry     4 June 80
0000     49 ;       Deleted BUS/PRB.RD.PTE from macro CLRTB.VA_VA and CLRCH.VA_VA to
0000     50 ;       satisfy validity checks.
0000     51 ;
0000     52 ; 14    Dave Spain      23 May 80
0000     53 ;       Added BUS/PRB.RD.PTE to the clear cache and clear TB macros for Mr.
0000     54 ;       Bill.
0000     55 ;
0000     56 ; 13    Dave Spain      14 Apr 80
0000     57 ;       Added SIZE[] or VSIZE/1 to some macros. This allows version 65 of
```

```
0000     58 ;       the COMET micro-code defines to run.
0000     59 ;
0000     60 ; 12    Bill Landry      1 Apr 80
0000     61 ;       Added BUS/PRB.RD micro order to RDM_TB macro to compensate for timing
0000     62 ;       problem in ADD gate array.  This is not an APRIL FOOL!
0000     63 ;
0000     64 ; 11    Dave Spain      27 Mar 80
0000     65 ;       Fixed TB macro's to include MSRC/VA to prevent validity failure.
0000     66 ;
0000     67 ; 10    Bill Landry     19 Mar 80
0000     68 ;       Added Q_R[].RL.P and VA_Q.OR.R[]
0000     69 ;
0000     70 ; 09    Bill Landry     19 Mar 80
0000     71 ;       Deleted TB_R[].RL.P.OR.D
0000     72 ;
0000     73 ; 08    Bill Landry     18 Mar 80
0000     74 ;       Added VA_R[].RL.P
0000     75 ;
0000     76 ; 07    Bill Landry     07 Mar 80
0000     77 ;       Added Q_M[].RL.PTE
0000     78 ;
0000     79 ; 06    Dave Spain      06 Mar 80
0000     80 ;       Changed TB_R[].RL.P macro to TB_R[].RL.P.OR.D.
0000     81 ;
0000     82 ; 05    Dave Spain      03 Mar 80
0000     83 ;       Added TB_R[].RL.P macro for Bill Landry.
0000     84 ;
0000     85 ; 04    Dave Spain      22 Feb 80
0000     86 ;       Added CLRTB.VA_R[] macro for Bill Landry.
0000     87 ;
0000     88 ; 03    Dave Spain      30 Jan 80
0000     89 ;       Added Tom Groetzinger's macro definitions.
0000     90 ;
0000     91 ; 02    Dave Spain      30 Jan 80
0000     92 ;       Added Tony Vezza's macro definitions.
0000     93 ;
0000     94 ; 01    Dave Spain      29 Jan 80
0000     95 ;       Macros initiated.
0000     96 ;--
0000     97
0000     98          .SBTTL  "       RDM Control File Macros"
0000     99
0000    100 ;STROBE.V.BUS            "RDMCF0/V.STROBE"
0000    101 ;DCS.ADDR_0              "RDMCF1/DCS.ADDR.CLR"
0000    102 ;STOP.CLOCK              "RDMCF2/STOP.CPU.CLOCK"
0000    103 ;LATCH.CS.ADDR           "RDMCF3/DIS.STROBE.CS.A"
0000    104 ;RDM_WB                  "RDMCF4/H.REG.FROM.WB"
0000    105 ;WB_RDM                  "RDMCF5/WB.FROM.D.REG"
0000    106 ;RDM_CMI                          "RDMCF6/STROBE.CMI"
0000    107 ;INH.CLOCK.SA            "RDMCF7/INH.SA.CLOCK"
0000    108
0000    109          .SBTTL  "       Diagnostic Macros"
0000    110
0000    111 ;CLOCK.EXT                        "CLKX/XTND"
0000    112 ;CLRCH.VA_RDM                     "WB_RDM,WCTRL/CLRCH.VA_WB,BUS/PRB.RD.PTE"
```

```
0000   113 ;CLRCH.VA_VA                    "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRCH.VA_WB
0000   114 ;CLRTB.VA_D                     "RSRC/ZERO,MUX/D.R1,ALU/OR,WCTRL/CLRTB.VA_WB,
                                                BUS/PRB.RD.PTE"
0000   115 ;
0000   116 ;CLRTB.VA_RDM                   "WB_RDM,WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
0000   117 ;CLRTB.VA_R[]                   "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/CLRTB.VA_WB,
                                                BUS/PRB.RD.PTE"
0000   118 ;
0000   119 ;CLRTB.VA_VA                    "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRTB.VA_WB
0000   120 ;CLRTB.VA_WB                    "WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
0000   121 ;D_LONLIT                       "RSRC/LONLIT,ALPCTL/WX_D_R.Q_D"
0000   122 ;D_M[]_LONLIT                   "D_LONLIT,MSRC/a1"
0000   123 ;D_VA_0                         "RSRC/ZERO,ROT/ZERO,MUX/R.S,ALU/OR,DQ1/D_WX,
                                                WCTRL/VA_WB"
0000   124 ;
0000   125 ;FPA_M[] FPA_RDM                        "MSRC/a1,WB_RDM,FPA/FPA_MBUS.FPA_WBUS"
0000   126 ;FPA_M[] FPA_R[]                        "FPA/FPA_MBUS.FPA_WBUS,MSRC/a1,RSRC/a2,MUX/R
                                                ROT/ZERO,ALU/OR"
0000   127 ;
0000   128 ;FPA_RDM                              "WB_RDM,FPA/FPA_DATA.WBUS"
0000   129 ;FPA_R[]                              "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,FPA/FPA_DAT
0000   130 ;FPA_R[]+M[]                    "RSRC/a1,MSRC/a2,MUX/M.R1,ALU/A+B+CI,ALUCI/ZERO,FPA/
0000   131 ;FPA_WB                        "FPA/FPA_DATA.WBUS"
0000   132 ;MEMSCAR_R[]                   "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/MEMSCAR_WB"
0000   133 ;M[]_FPA                              "MSRC/a1,SPW/MLONG,FPA/WBUS_FPA"
0000   134 ;M[]_LONLIT                    "MSRC/a1,SPW/MLONG,ALPCTL/WX_R.Q_D,RSRC/LONLIT"
0000   135 ;M[]_R[].OR.NIB0(M)            "MSRC/a1,RSRC/a2,SPW/MLONG,ROT/GETNIB,ALU/OR,
                                                MUX/R.S,DQ1/NOP"
0000   136 ;
0000   137 ;M[]_RDM                              "WB_RDM,SPW/MLONG,MSRC/a1"
0000   138 ;PC_PC+RDM                    "WCTRL/PC_PC+WB,WB_RDM"
0000   139 ;PC_PC+4 MB_XB                "MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,SIZE[LONG]"
0000   140 ;PC_PC+4 RDM_XB               "RSRC/ZERO,MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,
                                                MUX/M.R1,ALU/OR,RDM_WB,SIZE[LONG]"
0000   141 ;
0000   142 ;PC_RDM                       "WB_RDM,WCTRL/PC_WB"
0000   143 ;PSL_RDM                              "CCPSL/PSL_WB.CCBR_ALUS,WB_RDM"
0000   144 ;PSL_WB                       "CCPSL/PSL_WB.CCBR_ALUS"
0000   145 ;Q_D WX_R                     "ALPCTL/WX_R.Q_D"
0000   146 ;Q_LONLIT                     "RSRC/LONLIT,ALPCTL/WX_S.Q_R"
0000   147 ;Q_M[].RL.PTE                 "ALPCTL/WX_Q_S,MSRC/a1,ROT/RL.MM.PTE"
0000   148 ;Q_R[].AND.Q                  "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/AND"
0000   149 ;Q_R[].ASL.P                  "ALPCTL/WX_Q_S,RSRC/a1,ROT/ASL.R.P"
0000   150 ;Q_R[].OR.Q                   "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/OR"
0000   151 ;Q_R[].RL.P                   "ALPCTL/WX_Q_S,RSRC/a1,ROT/RL.RR.P"
0000   152 ;RDM_ALUF                     "ALPCTL/WB_ALUF,RDM_WB"
0000   153 ;RDM_FPA                              "FPA/WBUS_FPA,RDM_WB"
0000   154 ;RDM_LONLIT                   "WB_LONLIT,RDM_WB"
0000   155 ;RDM_LONLIT.Q_M               "RSRC/LONLIT,ALPCTL/WX_R.Q_M,RDM_WB"
0000   156 ;RDM_M[]                              "MSRC/a1,ROT/ZERO,MUX/M.S,ALU/OR,RDM_WB"
0000   157 ;RDM_M[].OR.R[]               "WB_M[a1].OR.R[a2],RDM_WB"
0000   158 ;RDM_M[]-R[]                  "WB_M[a1]-R[a2],RDM_WB"
0000   159 ;RDM_M[].XZ.OR.R[]           "ALU/A+B+CI,MUX/R.S,ROT/XZ.MM,MSRC/a1,RSRC/a2,
                                        RDM_WB"
0000   160 ;
0000   161 ;RDM_PC                       "MSRC/PC,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
0000   162 ;RDM_PCBACK                   "RSRC/ZERO,MSRC/PCBACK,MUX/M.R1,ALU/OR,RDM_WB"
0000   163 ;RDM_Q                        "RSRC/ZERO,MUX/R.Q,ALU/OR,RDM_WB"
0000   164 ;RDM_Q Q_D                    "ALPCTL/WX_Q.Q_D,RDM_WB"
0000   165 ;RDM_R[]                              "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,RDM_WB"
0000   166 ;RDM_TB                       "WB_M[TB],RDM_WB,BUS/PRB.RD,SIZE[LONG]"
0000   167 ;RDM_VA                       "MSRC/VA,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
```

```
0000   168 ;RNUM_LONLIT                    "ALPCTL/WX_D_R.Q_M,MSRC/RNUM_WBUS,RSRC/LONLIT"
0000   169 ;R[]_RDM                                "WB_RDM,SPW/RLONG,RSRC/a1"
0000   170 ;R[].DT_Q                       "RSRC/a1,D_Q_Q_D,SPW/RSIZE,VSIZE/1"
0000   171 ;R[]_VA                        "RSRC/a1,MSRC/VA,ROT/ZERO,MUX/M.S,ALU/OR,SPW/RLONG"
0000   172 ;R[]_WB                        "RSRC/a1,SPW/RLONG"
0000   173 ;TB_RDM                        "WCTRL/TB_WB,WB_RDM,MSRC/VA"
0000   174 ;TB_WB                         "WCTRL/TB_WB,MSRC/VA"
0000   175 ;TESTFPA []                    "TESTFPA/a1"
0000   176 ;VA_PC+LONLIT+I.PC_PC+I        "RSRC/LONLIT,MSRC/XB.PC_PC+I,ROT/ZERO,MUX/R.S,ALU/OR
0000   177 ;                                      ISTRM/ISIZE_DSIZE,WCTRL/VA_PC+I+W.PC_PC+I,
0000   178 ;                                      SIZE[LONG]"
0000   179 ;VA_Q.OR.R[]                    "WCTRL/VA_WB,RSRC/a1,ALU/OR,MUX/R.Q"
0000   180 ;VA_RDM                        "WB_RDM,WCTRL/VA_WB"
0000   181 ;VA_R[].RL.P                   "ALPCTL/WX_S,ROT/RL.RR.P,RSRC/a1,WCTRL/VA_WB"
0000   182 ;VA_VA-ZLIT0[]                 "WCTRL/VA_WB,LIT/LITRL,LITRL/a1,ROT/ZLIT0,MSRC/VA,
0000   183 ;                                      MUX/M.S,ALU/A-B-CI"
0000   184 ;VA_WB                         "WCTRL/VA_WB"
0000   185 ;WB_FPA CCR                    "FPA/WBUS_FPA.CC"
0000   186 ;WB_FPA TCR                    "WCTRL/FPTCR"
0000   187 ;WB_LONLIT                     "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
0000   188 ;WB_LONLIT.Q_M                 "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
0000   189 ;WB_M[]+R[]                    "ALU/A+B+CI,ALUCI/ZERO,MUX/M.R1,MSRC/a1,RSRC/a2"
0000   190 ;WB_R[].Q_D                    "RSRC/a1,ALPCTL/WX_R.Q_D"
0000   191 ;WDR_R[]                              "WCTRL/WDR_WB,ALU/OR,MUX/R.S,RSRC/a1,ROT/ZER
0000   192 ;WRITE.LONG R[]                "BUS/WRITE.LNG,WCTRL/WDR_WB.UR,RSRC/a1,ROT/ZERO,
0000   193 ;                                      MUX/R.S,ALU/OR,SIZE[LONG]"
0000   194         .SBTTL  ""
0000   195         .SBTTL  "MODULE GATE ARRAY MAPS"
```

```
                                        .SBTTL  "       L0001 MULTIPLIER AND GATE ARRAY LOCATION"
```

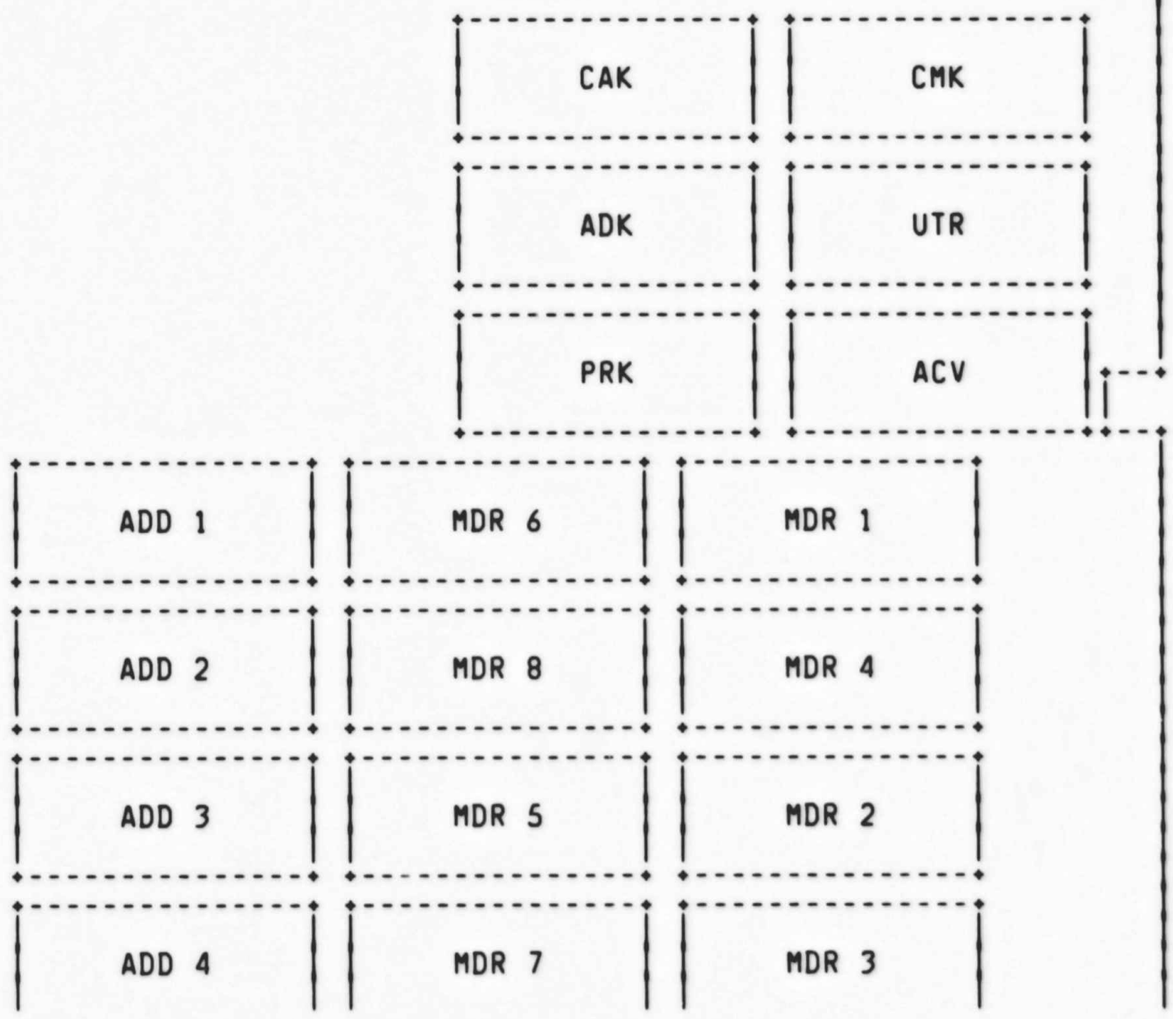


```
                0000   252 ;+-----------------------------------------------------------------------------+
```

```
0000   .SBTTL  "      L0002 GATE ARRAY LOCATION"
0000 ;+-----------------------------------------------------------------------+
0000 ;|                                                                       |
0000 ;|                                                                       |
0000 ;|                                                                       |
0000 ;|                                                                       |
0000 ;|                                                                       |
0000 ;|                                                                       |
0000 ;|                                                                       |
0000 ;|                                                                       |
0000 ;|                                                                       |
0000 ;|                                                                       |
0000 ;|                    +---------------+ +---------------+                |
0000 ;|                    |               | |               |                |
0000 ;|                    |      PHB      | |      MSQ      |                |
0000 ;|                    |               | |               |                |
0000 ;|                    +---------------+ +---------------+                |
0000 ;|                    +---------------+ +---------------+            +---+
0000 ;|                    |               | |               |            |   |
0000 ;|                    |      CCC      | |      SAC      |            +---+
0000 ;|                    |               | |               |                |
0000 ;|                    +---------------+ +---------------+                |
0000 ;|  +---------------+ +---------------+ +---------------+                |
0000 ;|  |               | |               | |               |                |
0000 ;|  |      SPA      | |      SRK      | |      IRD      |                |
0000 ;|  |               | |               | |               |                |
0000 ;|  +---------------+ +---------------+ +---------------+                |
0000 ;|  +---------------+ +---------------+ +---------------+                |
0000 ;|  |               | |               | |               |                |
0000 ;|  |      TOK      | |      CLA      | |      ALK      |                |
0000 ;|  |               | |               | |               |                |
0000 ;|  +---------------+ +---------------+ +---------------+                |
0000 ;|                    +---------------+ +---------------+                |
0000 ;|                    |               | |               |                |
0000 ;|                    |      ALP 1    | |      ALP 5    |                |
0000 ;|                    |               | |               |                |
0000 ;|                    +---------------+ +---------------+            +---+
0000 ;|                    +---------------+ +---------------+            |   |
0000 ;|                    |               | |               |            +---+
0000 ;|                    |      ALP 2    | |      ALP 6    |                |
0000 ;|                    |               | |               |                |
0000 ;|                    +---------------+ +---------------+                |
0000 ;|                    +---------------+ +---------------+                |
0000 ;|                    |               | |               |                |
0000 ;|                    |      ALP 3    | |      ALP 7    |                |
0000 ;|                    |               | |               |                |
0000 ;|                    +---------------+ +---------------+                |
0000 ;|                    +---------------+ +---------------+                |
0000 ;|                    |               | |               |                |
0000 ;|                    |      ALP 4    | |      ALP 8    |                |
0000 ;|                    |               | |               |                |
0000 ;|                    +---------------+ +---------------+                |
0000 ;|  +---------------+ +---------------+ +---------------+ +---------------+
0000 ;|  |               | |               | |               | |               |
0000 ;|  |      SRM 1    | |      SRM 2    | |      SRM 3    | |      SRM 4    |
0000 ;|  |               | |               | |               | |               |
0000 ;|  +---------------+ +---------------+ +---------------+ +---------------+
```

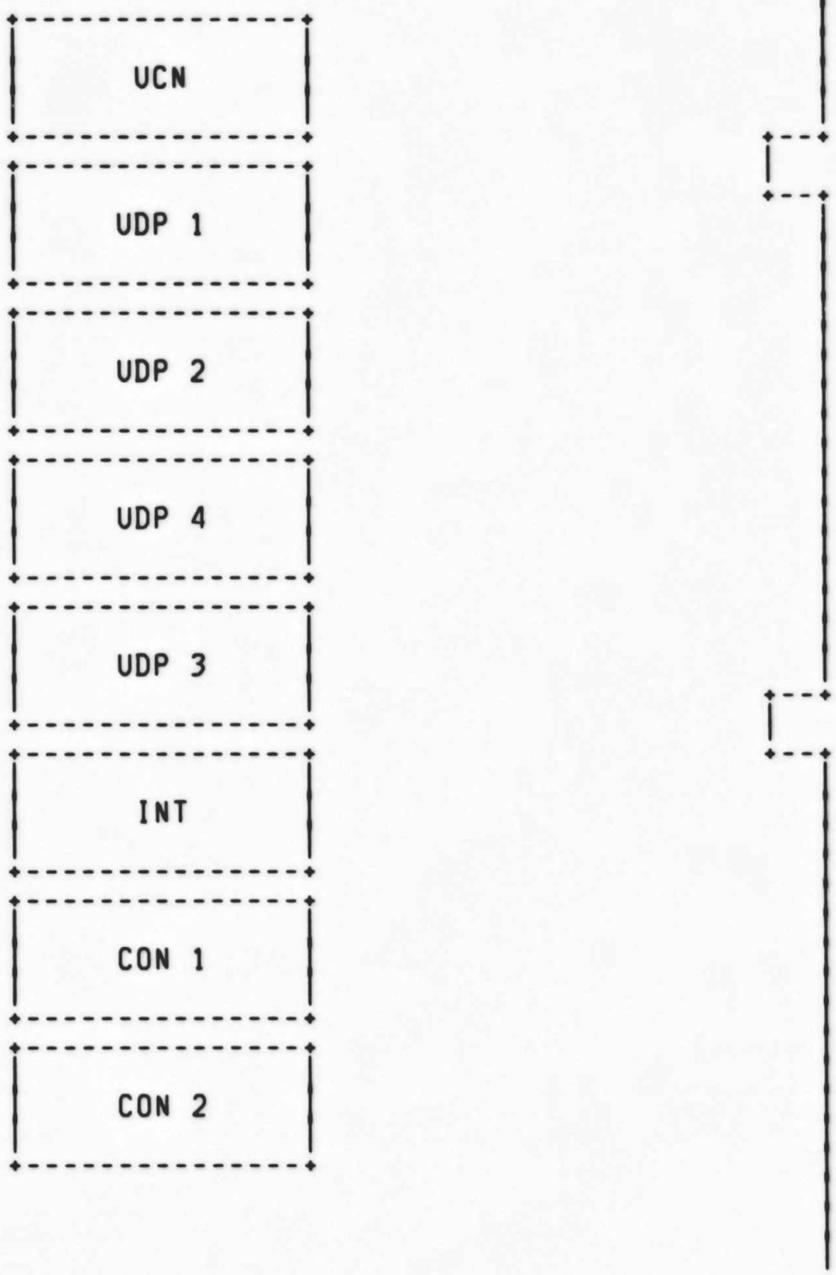
```
                0000   309 ;+-----------------------------------------------------------------------------+
```

```
                        0000    311          .SBTTL  "       L0003 GATE ARRAY LOCATION"
                        0000    312 ;+-----------------------------------------------------------------------+
                        0000    313 ;|                                                                       |
                        0000    314 ;|                                                                       |
                        0000    315 ;|                                                                       |
                        0000    316 ;|                                                                       |
                        0000    317 ;|                                                                       |
                        0000    318 ;|                                                                       |
                        0000    319 ;|                                                                       |
                        0000    320 ;|                                                                       |
                        0000    321 ;|                                                                       |
                        0000    322 ;|                                                                       |
                        0000    323 ;|                                                                       |
                        0000    324 ;|                                                                       |
                        0000    325 ;|                                                                       |
                        0000    326 ;|                                                                       |
                        0000    327 ;|                                                                       |
                        0000    328 ;|                                                                       |
                        0000    329 ;|                                                                       |
                        0000    330 ;|                                                                       |
                        0000    331 ;|                                                                       |
                        0000    332 ;|                                                                       |
                        0000    333 ;|                                                                       |
                        0000    334 ;|                                                                       |
                        0000    335 ;|                                                                       |
                        0000    336 ;|                                                                       |
                        0000    337 ;|                                                                       |
                        0000    338 ;|                                                                       |
                        0000    339 ;|                                                                       |
                        0000    340 ;|                                                                       |
                        0000    341 ;|                                                                       |
                        0000    342 ;|                                                                       |
                        0000    343 ;|                                                                       |
                        0000    344 ;|                                                                       |
                        0000    345 ;|                                                                       |
                        0000    346 ;|                                                                       |
                        0000    347 ;|                                                                       |
                        0000    348 ;|                                                                       |
                        0000    349 ;|                                                                       |
                        0000    350 ;|                                                                       |
                        0000    351 ;|                                                                       |
                        0000    352 ;|                                                                       |
                        0000    353 ;|                                                                       |
                        0000    354 ;|                                                                       |
                        0000    355 ;|                                                                       |
                        0000    356 ;|                                                                       |
                        0000    357 ;|                                                                       |
                        0000    358 ;|                                                                       |
                        0000    359 ;|                                                                       |
                        0000    360 ;|                                                                       |
                        0000    361 ;|                                                                       |
                        0000    362 ;|                                                                       |
                        0000    363 ;|                                                                       |
                        0000    364 ;|                                                                       |
                        0000    365 ;|                                                                       |
```



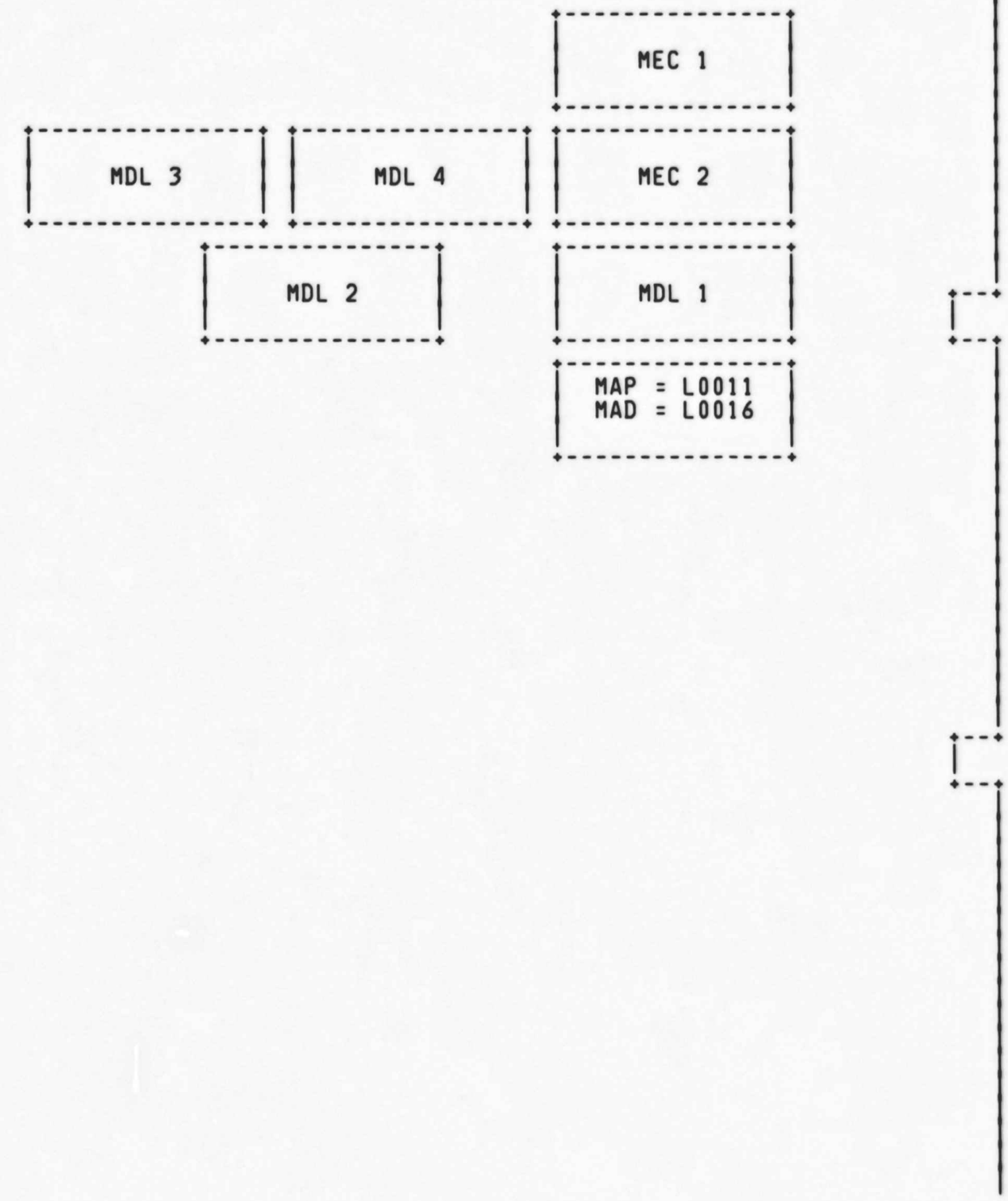
```
ZZ-ECKAB-9.2    " L0003 GATE ARRAY LOCATION"                    13-MAY-1987       Fiche 4 Frame K10          Sequence 745
ECKAB                    VAX-11/750 DPM MICRO DIAGNOSTICS        11-DEC-1985 09:45:22  VAX/VMS Macro V04-00        Page  1
V09.01                   " L0003 GATE ARRAY LOCATION"            14-AUG-1984 15:55:16  [VAX750]COMETASM.MAR;221          (1

                    0000   366 ;|              +---------------+ +---------------+ +---------------+          |
                    0000   367 ;+-----------------------------------------------------------------------------+
```

```
                        0000    369          .SBTTL  "       L0004 GATE ARRAY LOCATION"
                        0000    370 ;+----------------------------------------------------------------------------+
                        0000    371 ;|                                                                            |
                        0000    372 ;|                                                                            |
                        0000    373 ;|                                                                            |
                        0000    374 ;|                                                                            |
                        0000    375 ;|                                                                            |
                        0000    376 ;|                                                                            |
                        0000    377 ;|                                                                            |
                        0000    378 ;|                                                                            |
                        0000    379 ;|                                                                            |
                        0000    380 ;|                                                                            |
                        0000    381 ;|                                                                            |
                        0000    382 ;|                                                                            |
                        0000    383 ;|                                                                            |
                        0000    384 ;|                                                                            |
                        0000    385 ;|                                                                            |
                        0000    386 ;|                                                                            |
                        0000    387 ;|                                                                            |
                        0000    388 ;|                                                                            |
                        0000    389 ;|                                                                            |
                        0000    390 ;|                                                                            |
                        0000    391 ;|                                                                            |
                        0000    392 ;|                                                                            |
                        0000    393 ;|                                                                            |
                        0000    394 ;|                                                                            |
                        0000    395 ;|                                                                            |
                        0000    396 ;|                                                                            |
                        0000    397 ;|                                                                            |
                        0000    398 ;|                                                                            |
                        0000    399 ;|                                                                            |
                        0000    400 ;|                                                                            |
                        0000    401 ;|                                                                            |
                        0000    402 ;|                                                                            |
                        0000    403 ;|                                                                            |
                        0000    404 ;|                                                                            |
                        0000    405 ;|                                                                            |
                        0000    406 ;|                                                                            |
                        0000    407 ;|                                                                            |
                        0000    408 ;|                                                                            |
                        0000    409 ;|                                                                            |
                        0000    410 ;|                                                                            |
                        0000    411 ;|                                                                            |
                        0000    412 ;|                                                                            |
                        0000    413 ;|                                                                            |
                        0000    414 ;|                                                                            |
                        0000    415 ;|                                                                            |
                        0000    416 ;|                                                                            |
                        0000    417 ;|                                                                            |
                        0000    418 ;|                                                                            |
                        0000    419 ;|                                                                            |
                        0000    420 ;|                                                                            |
                        0000    421 ;|                                                                            |
                        0000    422 ;|                                                                            |
                        0000    423 ;|
```



```
ZZ-ECKAB-9.2    " L0004 GATE ARRAY LOCATION"                    13-MAY-1987       Fiche 4 Frame M10          Sequence 747
ECKAB                      VAX-11/750 DPM MICRO DIAGNOSTICS       11-DEC-1985 09:45:22  VAX/VMS Macro V04-00        Page  1
V09.01                     " L0004 GATE ARRAY LOCATION"           14-AUG-1984 15:55:16  [VAX750]COMETASM.MAR;221          (1

                    0000    424 ;|                                                                                    |
                    0000    425 ;+------------------------------------------------------------------------------------+
```

```
                        .SBTTL  "       L0011 & L0016 GATE ARRAY LOCATION"
```



```
0000   482 ;|                                                                    |
0000   483 ;+-------------------------------------------------------------------+
```

```
                     0000   485          .SBTTL  ""
                     0000   486          .LIST   MC
                     0000   487          .LIBRARY        /UCOD$0:[VAX750]COMETMAC/
                     0000   488          .LIST   ME
                     0000   489          .NLIST  CND,MC
                     0000                .SBTTL  "EQUATED SYMBOLS
                     0000        ;
                     0000        ; RDM REGISTER DEFINITIONS:
                     0000        ;
                     0000        ; NOTE: THESE REGITER DEFINITIONS ARE OFFSET BY THE LOAD/START ADDRESS OF
                     0000        ;       THE PROGRAM. FOR THE SIMULATOR VERSION THIS IS 400(X) AND FOR THE
                     0000        ;       REAL VERSION IT IS 8400(X).
                     0000        ;
00007400             0000                DCS_DATA_REG     =       ^XF800 + HEAD
00007401             0000                DCS_ADDR_REG_WO  =       ^XF801 + HEAD
00007402             0000                DCS_CTRL_REG     =       ^XF802 + HEAD
00007403             0000                DCS_CTRL_FILE    =       ^XF803 + HEAD
00007404             0000                ADDR_MATCH_LO    =       ^XF804 + HEAD
00007405             0000                ADDR_MATCH_HI    =       ^XF805 + HEAD
00007406             0000                STATUS_REG       =       ^XF806 + HEAD
                     0000        ;
00007410             0000                A_REG_BYTE_0     =       ^XF810 + HEAD
00007411             0000                A_REG_BYTE_1     =       ^XF811 + HEAD
00007412             0000                A_REG_BYTE_2     =       ^XF812 + HEAD
00007413             0000                A_REG_BYTE_3     =       ^XF813 + HEAD
00007414             0000                MD_REG_BYTE_0    =       ^XF814 + HEAD
00007415             0000                MD_REG_BYTE_1    =       ^XF815 + HEAD
00007416             0000                MD_REG_BYTE_2    =       ^XF816 + HEAD
00007417             0000                MD_REG_BYTE_3    =       ^XF817 + HEAD
00007418             0000                MH_REG_BYTE_0    =       ^XF818 + HEAD
00007419             0000                MH_REG_BYTE_1    =       ^XF819 + HEAD
0000741A             0000                MH_REG_BYTE_2    =       ^XF81A + HEAD
0000741B             0000                MH_REG_BYTE_3    =       ^XF81B + HEAD
0000741C             0000                CMI_CTRL_REG     =       ^XF81C + HEAD
0000741D             0000                MAINT_REG        =       ^XF81D + HEAD
0000741E             0000                RD_CTRL_REG      =       ^XF81E + HEAD
                     0000        ;
00007420             0000                FRONT_PNL_1      =       ^XF820 + HEAD
00007421             0000                FRONT_PNL_2      =       ^XF821 + HEAD
00007422             0000                CS_ADD_TRAP_LOW  =       ^XF822 + HEAD
00007423             0000                CS_ADD_TRAP_HGH  =       ^XF823 + HEAD
00007424             0000                CS_ADD_BUFF_LOW  =       ^XF824 + HEAD
00007425             0000                CS_ADD_BUFF_HGH  =       ^XF825 + HEAD
00007426             0000                BUFF_ADDR_LOW    =       ^XF826 + HEAD
00007427             0000                DCS_ADDR_REG_RO  =       ^XF827 + HEAD
                     0000        ;
0000742C             0000                WD_REG_BYTE_0    =       ^XF82C + HEAD
0000742D             0000                WD_REG_BYTE_1    =       ^XF82D + HEAD
0000742E             0000                WD_REG_BYTE_2    =       ^XF82E + HEAD
0000742F             0000                WD_REG_BYTE_3    =       ^XF82F + HEAD
00007430             0000                WH_REG_BYTE_0    =       ^XF830 + HEAD
00007431             0000                WH_REG_BYTE_1    =       ^XF831 + HEAD
00007432             0000                WH_REG_BYTE_2    =       ^XF832 + HEAD
00007433             0000                WH_REG_BYTE_3    =       ^XF833 + HEAD
                     0000        ;
```

```
                        0000        ; RDM REGISTER NAMES AS DEFINED IN HARDWARE SPEC
                        0000        ;
                00007400 0000               DCSDA=DCS_DATA_REG              ;Diagnostic control store data register
                00007401 0000               DCSAD=DCS_ADDR_REG_WO           ;Diagnostic control store address reg
                00007402 0000               DCSCR=DCS_CTRL_REG              ;Diagnostic control store control reg
                00007403 0000               DCSCF=DCS_CTRL_FILE             ;Diagnostic control store control file
                00007404 0000               CSAMR=ADDR_MATCH_LO             ;Control store address match register
                00007406 0000               STATUS=STATUS_REG               ;Status register
                00007410 0000               MAREG=A_REG_BYTE_0              ;Memory address register
                00007414 0000               MDREG=MD_REG_BYTE_0             ;Memory data register
                00007418 0000               MHREG=MH_REG_BYTE_0             ;Memory holding register
                0000741C 0000               CMICTL=CMI_CTRL_REG             ;CMI control
                0000741D 0000               MAIN32=MAINT_REG                ;32 Bit path maintenance control
                0000741E 0000               RDCTRL=RD_CTRL_REG              ;Remote diagnosis control
                00007420 0000               FRONT1=FRONT_PNL_1              ;Front panel readback 1
                00007421 0000               FRONT2=FRONT_PNL_2              ;Front panel readback 2
                00007422 0000               CSTRP=CS_ADD_TRAP_LOW           ;Control store address trap register
                00007424 0000               CSBUF=CS_ADD_BUFF_LOW           ;Control store address buffer
                00007426 0000               BADD1=BUFF_ADDR_LOW             ;Control store address trace readback
                00007427 0000               CLKDCS=DCS_ADDR_REG_RO          ;Clock control & DCS address register
                0000742C 0000               WDREG=WD_REG_BYTE_0             ;WBUS data register
                00007430 0000               WHREG=WH_REG_BYTE_0             ;WBUS holding register
                        0000        ;
                        0000        ; RDM REGISTER BIT DEFINITIONS:
                        0000        ;
                        0000        ; DCS CONTROL REGISTER
                00000001 0000               HALT_ON_MATCH    =      1
                00000002 0000               CLEAR_CTRL_FILE  =      2
                00000004 0000               TRACE_ENABL      =      4
                00000008 0000               PAR_CHK_ENABL    =      8
                00000010 0000               PAR_STOP_ENABL   =      ^X10
                00000020 0000               CLK_CTRL_0       =      ^X20    ; ACTIVE LOW
                00000040 0000               CLK_CTRL_1       =      ^X40    ; ACTIVE LOW
                00000080 0000               MICRO_ADDR_INH   =      ^X80
                        0000            ;
                        0000            ; FOLLOWING IS A FUNCTION TABLE OF THE CLOCK CONTROL BITS
                        0000            ;
                        0000                ;       CLK_CTRL 1,0    FUNCTION
                        0000                ;           11            HALT
                        0000                ;           01            SINGLE MICRO INSTRUCTION
                        0000                ;           10            SINGLE TICK
                        0000                ;           00            RUN
                        0000        ;
                        0000        ; DCS CONTROL FILE      ALL OF THESE BITS ARE ACTIVE LOW
                        0000        ;
                00000001 0000               V_BUS_STROBE     =      1
                00000002 0000               DCS_ADDR_CLEAR   =      2
                00000004 0000               STOP_CLOCK       =      4
                00000008 0000               EN_TRACE_REG     =      8
                00000010 0000               H_FROM_WBUS      =      ^X10
                00000020 0000               WBUS_FROM_D      =      ^X20
                00000040 0000               STROBE_CMI       =      ^X40
                00000080 0000               SA_CLOCK         =      ^X80
                        0000        ;
                        0000        ; CS ADDRESS MATCH REGISTER (HIGH)
```

```
                    0000        ;
00000040            0000                VBUS_SERIAL_IN  =       ^X40    ; ACTIVE LOW
00000080            0000                VBUS_CLOCK      =       ^X80
                    0000        ;
                    0000        ; STATUS REGISTER
                    0000        ;
00000001            0000                VBUS_SERIAL_OUT =       1
00000002            0000                TRAP_OFF        =       2
00000008            0000                CMI_STATUS_0    =       8
00000010            0000                CMI_STATUS_1    =       ^X10
                    0000                ;
                    0000                ; THE CMI STATUS BITS ARE ENCODED AS FOLLOWS:
                    0000                ;
                    0000                ;       00 = NO RESPONSE
                    0000                ;       01 = UNCORRECTABLE ERROR
                    0000                ;       10 = CORRECTABLE ERROR
                    0000                ;       11 = NO ERRORS
00000020            0000                CMI_COMPLETE    =       ^X20
                    0000        ;
00000080            0000                CLOCK_STOPED    =       ^X80
                    0000        ;
                    0000        ; CMI CONTROL REGISTER
                    0000        ;
00000001            0000                CMI_CTRL_CLEAR  =       1
00000008            0000                CMI_GO          =       8
00000020            0000                CMI_WRITE       =       ^X20
00000040            0000                CMI_READ        =       ^X40
00000080            0000                SA_START_STOP   =       ^X80
                    0000        ;
                    0000        ; 32 BIT MAINTENANCE CONTROL REGISTER
                    0000        ;
00000001            0000                MAINT_TRAP_OFF  =       1
00000002            0000                MAINT_STB_INH   =       2
00000004            0000                MAINT_DCS_ENABL =       4
00000008            0000                MAINT_WB_TO_WH  =       8
00000010            0000                MAINT_WD_TO_WB  =       ^X10
00000020            0000                MAINT_CMI_TO_MH =       ^X20
00000040            0000                MAINT_MD_TO_CMI =       ^X40
00000080            0000                MAINT_A_TO_CMI  =       ^X80
                    0000        ;
                    0000        ; RD CONTROL REGISTER
                    0000        ;
00000008            0000                MASTER_HALT_EN  =       8
00000010            0000                VBUS_LOAD       =       ^X10
00000020            0000                TRAP_HALT_EN    =       ^X20
00000040            0000                DC_LOW          =       ^X40
00000080            0000                AC_LOW          =       ^X80
                    0000        ;
                    0000        ; FRONT PANEL REGISTER 1
                    0000        ;
00000001            0000                FP_BOOT_0       =       1
00000002            0000                FP_BOOT_1       =       2
00000004            0000                FP_START_0      =       4
00000008            0000                FP_START_1      =       8
00000010            0000                CPU_RUN         =       ^X10
```

```
00000020  0000                PARITY_ERROR    =       ^X20
          0000          ;
          0000          ; FRONT PANEL REGISTER 2
          0000          ;
00000001  0000                REMOTE          =       1
00000002  0000                FAULT           =       2
00000004  0000                TEST            =       4
          0000          ;
          0000          ;
          0000          ;
00000040  0000                PB_INIT         =       ^X40    ; ACTIVE LOW
00000080  0000                FRONT_PNL_LOCK  =       ^X80
          0000          ;
          0000          ; DCS ADDRESS REGISTER READ ONLY
          0000          ;
00000040  0000                CLK_CTRL_0_RO   =       ^X40    ; ACTIVE HIGH
00000080  0000                CLK_CTRL_1_RO   =       ^X80    ; ACTIVE HIGH
          0000          ;
          0000          ; MISCELLANEOUS EQUATES:
          0000          ;
00000001  0000                BYTE            =       1
00000002  0000                WORD            =       2
00000004  0000                LONG            =       4
0000000A  0000                MICROWORD       =       10
00002C5E  0000                MONITOR_SIZE    =       ^X2C5E
00000400  0000                STACK_SIZE      =       1024
000007A2  0000                M$TEST_SIZE     =       14336 - STACK_SIZE - MONITOR_SIZE
          0000                                          ; MAXIMUM SIZE OF A TEST OVERLAY
000003E2  0000                M$TEST_SIZE_D   = M$TEST_SIZE - 960
          0000                                          ; MAXIMUM SIZE OF TEST OVERLAY WITH
          0000                                          ; 'DEBUG' ENABLED IN
00000001  0000                B               =       BYTE
00000002  0000                W               =       WORD
00000004  0000                L               =       LONG
0000000A  0000                M               =       MICROWORD
00000001  0000                HIGH            =       1
00000000  0000                LOW             =       0
00000000  0000                ONERROR         =       0
00000001  0000                NOERROR         =       1
00000002  0000                ONEQUAL         =       2
00000003  0000                NOTEQUAL        =       3
00000010  0000                MAX_ARGUMENTS   =       16      ; BYTE LENGTH OF LARGEST PSEUDO OP
00000036  0000                DCS_SCRATCH_ADR =       54      ; DCS SCRATCH ADDRESS START
000032FC  0000                ERROR_LOG_ADR   =       ^XB6FC + HEAD
          0000                                          ; START ADDRESS OF MEMORY ERROR LOG
          0000    491   .NLIST  ME
          0000    492   .LIST   MC
          0000    493   .SBTTL  ""
          0000    494   .NLIST  MC      ; SHUT-OFF LISTING FOR FIRST 'NEWTEST' CALL
```

```
0011        .SBTTL  "TEST   08F     N BIT TEST 1
0011      3 ;*******************************************************************************
0011      4 ;++
0011      5 ;
0011      6 ; FUNCTIONAL DESCRIPTION:
0011      7 ;
0011      8 ;       THIS TEST VERIFIES THAT THE N BIT IN THE CCC GATE ARRAY SETS PROPERLY
0011      9 ;       FOR NATIVE MODE INSTRUCTIONS USING CC/CCOP2.CCBR_SIGND MICRO ORDER.
0011     10 ;
0011     11 ;
0011     12 ; TEST ALGORITHM:
0011     13 ;
0011     14 ;       1.  CREATE A TEST LOOP OF 36
0011     15 ;       2.  MODIFY THE MICRO CODE FOR EACH LOOP THROUGH THE TEST
0011     16 ;       3.  LOAD THE NEW MICRO CODE INTO THE RDM DCS
0011     17 ;       4.  SAVE THE LOOP COUNT
0011     18 ;       5.  MOVE THE LOOP COUNT TO THE RDM WDREG
0011     19 ;       6.  EXECUTE THE DCS PROGRAM
0011     20 ;               a.  DISABLE ^P INTERRUPTS
0011     21 ;               b.  SET THE IPL TO 1F AND MODE TO NATIVE
0011     22 ;               c.  MOVE THE LOOP COUNT TO THE RTEMPS
0011     23 ;               d.  LOAD THE PLATCH WITH 2
0011     24 ;               e.  ROTATE THE LOOP COUNT LEFT BY 2 TO FORM A CACHE ADDRESS
0011     25 ;               f.  LOAD THE PC WITH THE CACHE ADDRESS TO FORCE PREFETCH
0011     26 ;               g.  PERFORM IRD1TEST TO LOAD THE IR REGISTER
0011     27 ;               h.  LOAD THE Q REGISTER WITH TEST DATA
0011     28 ;               i.  ADD THE Q REGISTER TO ONE OF THE MTEMPS TO CREATE THE
0011     29 ;                       TEST CONDITIONS WHILE PERFORMING CCOP2.
0011     30 ;               j.  READ THE PSL TO THE RDM
0011     31 ;       7.  TEST THE PSL N BIT FOR CORRECT STATE
0011     32 ;       8.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST PATTERNS
0011     33 ;
0011     34 ;
0011     35 ; TEST PATTERNS:
0011     36 ;
0011     37 ;       ITERATION       IR      Q REG. + M[TEMP*]       SIZE
0011     38 ;
0011     39 ;           1           3F      80000000 + 00004000     LONG
0011     40 ;           2           41      000000C0 + 000000C0     BYTE
0011     41 ;           3           0E      00000080 + 00000000     BYTE
0011     42 ;           4           41      00000000 + 00000000     WORD
0011     43 ;           5           C0      00000080 + 00004040     BYTE
0011     44 ;           6           6C      00008000 + 00004040     LONG
0011     45 ;           7           F7      00000080 + 00004040     BYTE
0011     46 ;           8           41      00008000 + 00004040     BYTE
0011     47 ;           9           28      00000080 + 00004040     BYTE
0011     48 ;           A           5C      00000000 + 00004040     LONG
0011     49 ;           B           58      00008000 + 00004040     WORD
0011     50 ;           C           11      00000080 + 00004040     BYTE
0011     51 ;           D           10      00000080 + 00004040     BYTE
0011     52 ;           E           E5      00000080 + 00004040     BYTE
0011     53 ;           F           41      00000000 + 00000000     BYTE
0011     54 ;          10           41      00000000 + 00000000     LONG
0011     55 ;          11           78      00000080 + 00004040     BYTE
0011     56 ;          12           F4      00000080 + 00004040     BYTE
```

```
                0011    57 ;            13          48      00000080 + 00004040     BYTE
                0011    58 ;            14          74      00008000 + 00004040     LONG
                0011    59 ;            15          41      00008000 + 00008000     WORD
                0011    60 ;            16          C1      00000000 + 00000000     WORD
                0011    61 ;            17          41      80000000 + 80000000     LONG
                0011    62 ;            18          D8      00000080 + 00004040     BYTE
                0011    63 ;            19          0C      00000080 + 00004040     BYTE
                0011    64 ;            1A          96      00000080 + 00004040     BYTE
                0011    65 ;            1B          41      00004000 + 00000000     BYTE
                0011    66 ;            1C          EE      00000080 + 00004040     BYTE
                0011    67 ;            1D          F2      00008000 + 00004040     WORD
                0011    68 ;            1E          98      00000080 + 00004040     BYTE
                0011    69 ;            1F          0B      80000000 + 00004040     LONG
                0011    70 ;            20          53      00008000 + 00004040     LONG
                0011    71 ;            21          4F      00008000 + 00004040     LONG
                0011    72 ;            22          4C      00000080 + 00004040     BYTE
                0011    73 ;            23          A8      00000080 + 00004040     BYTE
                0011    74 ;            24          B0      00000080 + 00004040     BYTE
                0011    75 ;
                0011    76 ;
                0011    77 ; TEST ALGORITHM:
                0011    78 ;
                0011    79 ;        THESE TEST PATTERNS WERE GENERATED USING A LOGIC SIMULATOR.  SO IF
                0011    80 ;        YOU ATTEMPT TO READ THE VAX SRM AND TRY AN CORRELATE THE INSTRUCTIONS
                0011    81 ;        USED VERSUS CONDITIONS CODES IT REALLY WON'T MAKE ANY SENSE.  THE
                0011    82 ;        ATTEMPT HERE IS TO TEST THE INTERNAL WORKINGS OF THE CCC GATE ARRAY.
                0011    83 ;        ANY FAILUES SHOULD BE CONFINED TO THIS ARRAY OR THE INSTRUCTION
                0011    84 ;        REGISTER FEEDING IT.
                0011    85 ;
                0011    86 ; ASSUMPTIONS:
                0011    87 ;
                0011    88 ;        THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
                0011    89 ;
                0011    90 ; ERROR DESCRIPTION:
                0011    91 ;
                0011    92 ;        EXPECTED N BIT STATE
                0011    93 ;        RECEIVED N BIT STATE
                0011    94 ;        LOOP COUNT I
                0011    95 ;
                0011    96 ;--
                0011    97 ;*****************************************************************************
                0011    98
                0011    99
                0011   100          INITIALIZE                                  ;INT THE CPU
                0013   101          LOOP            I,1,36                      ;CREATE A TEST LOOP OF 36
                001A   102          LDFIELD         6$,I,B,4$,34,39,            ;MODIFY U-CODE FOR THIS LOOP
                002B   103          LDFIELD         7$,I,B,5$,40,41,            ;...
                003C   104          LDFIELD         8$,I,B,5$,64,68,            ;...
                004D   105          LOADDCS         4$,,09,2                    ;LOAD U-CODE INTO DCS
                0054   106          SAVEINDEX       I,3$                        ;SAVE LOOP COUNT
                0059   107          LOADREG         WDREG,3$,,L                 ;PUT LOOP COUNT IN WDREG
                0061   108          ERRLOOP                                     ;ERROR LOOP POINT
                0063   109          FETCH           0                           ;START DCS PROGRAM
                0068   110          BRSTCLK         <^X0C>                      ;END DCS PROGRAM
                006C   111          CMPREGMSK       WHREG,1$,I,2$,,L,           ;TEST FOR N BIT
```

```
                                0078          IFERROR        ,<CCC>                  ;BIT INCORRECT
                                007E          ENDLOOP        I                       ;END TEST LOOP
                                0081          SKIP                                   ;GO TO NEXT TEST
                                0088
                                0088
                                0088    BEGIN_TEST_DATA
                                0088
                                0088    ;-----------------------------------------------------------------------
                                0088
                        00000008  0088    1$:     .LONG   ^X00000008                     ;RESULTS TABLE
                        00000000  008C          .LONG   ^X00000000
                        00000000  0090          .LONG   ^X00000000
                        00000008  0094          .LONG   ^X00000008
                        00000008  0098          .LONG   ^X00000008
                        00000008  009C          .LONG   ^X00000008
                        00000000  00A0          .LONG   ^X00000000
                        00000000  00A4          .LONG   ^X00000000
                        00000000  00A8          .LONG   ^X00000000
                        00000000  00AC          .LONG   ^X00000000                     ;LOOP 10
                        00000008  00B0          .LONG   ^X00000008
                        00000000  00B4          .LONG   ^X00000000
                        00000008  00B8          .LONG   ^X00000008
                        00000000  00BC          .LONG   ^X00000000
                        00000008  00C0          .LONG   ^X00000008
                        00000008  00C4          .LONG   ^X00000008
                        00000008  00C8          .LONG   ^X00000008
                        00000008  00CC          .LONG   ^X00000008
                        00000000  00D0          .LONG   ^X00000000
                        00000008  00D4          .LONG   ^X00000008                     ;LOOP 20
                        00000000  00D8          .LONG   ^X00000000
                        00000000  00DC          .LONG   ^X00000000
                        00000000  00E0          .LONG   ^X00000000
                        00000008  00E4          .LONG   ^X00000008
                        00000008  00E8          .LONG   ^X00000008
                        00000008  00EC          .LONG   ^X00000008
                        00000000  00F0          .LONG   ^X00000000
                        00000008  00F4          .LONG   ^X00000008
                        00000008  00F8          .LONG   ^X00000008
                        00000008  00FC          .LONG   ^X00000008                     ;LOOP 30
                        00000008  0100          .LONG   ^X00000008
                        00000008  0104          .LONG   ^X00000008
                        00000008  0108          .LONG   ^X00000008
                        00000000  010C          .LONG   ^X00000000
                        00000008  0110          .LONG   ^X00000008
                        00000008  0114          .LONG   ^X00000008
                        00000008  0118    2$:     .LONG   ^X00000008                     ;MASK FOR CMPREGMSK PSEUDO OP
                        00000000  011C    3$:     .LONG   ^X00000000                     ;LOOP COUNT STORAGE
                                0120    4$:
U 09, 7700,0800,5BE5,0300,0470,180A  0120    ;% 09   Q_R[TEMP0]                             ;U-CODE THAT GETS MODIFIED FOR
                                012B    5$:                                            ;EACH LOOP
U 0A, 7701,8806,0090,0201,0470,180B  012B    ;% 0A   SIZE[LONG],WB_M[TEMP6]+Q,CCOP2         ;...
      03 02 04 02 04 02 03 02 01 00  0136    6$:     .BYTE   00,01,02,03,02,04,02,04,02,03  ;RSRC LDFIELD TABLE
      04 02 02 02 03 03 02 02 02 04  0140          .BYTE   04,02,02,02,03,03,02,02,02,04
      02 04 02 05 02 02 02 00 03 04  014A          .BYTE   04,03,00,02,02,02,05,02,04,02
                02 02 02 04 04 00  0154          .BYTE   00,04,04,02,02,02
```

```
                                        015A   171
      02 00 00 00 02 00 01 00 00 02     015A   172 7$:     .BYTE   02,00,00,01,00,02,00,00,00,02   ;SIZE LDFIELD TABLE
      02 00 00 00 02 00 00 00 00 01     0164   173         .BYTE   01,00,00,00,00,02,00,00,00,02
      00 01 00 00 00 00 00 02 01 01     016E   174         .BYTE   01,01,02,00,00,00,00,00,01,00
                  00 00 00 02 02 02     0178   175         .BYTE   02,02,02,00,00,00
                                        017E   176
      06 06 06 06 06 06 03 03 01 06     017E   177 8$:     .BYTE   06,01,03,03,06,06,06,06,06,06   ;MSRC LDFIELD TABLE
      06 06 06 06 03 03 06 06 06 06     0188   178         .BYTE   06,06,06,06,03,03,06,06,06,06
      06 06 06 03 06 06 06 00 03 04     0192   179         .BYTE   04,03,00,06,06,06,03,06,06,06
                  06 06 06 06 06 06     019C   180         .BYTE   06,06,06,06,06,06
                                        01A2   181
                                        01A2   182
                                        01A2   183 BEGIN_CPU_DATA
                                        0002   184
                                        0002   185 DCS_DATA
                                        0001   186
U 00, 7700,C800,0364,0300,0470,1801     0001   187 ;% 00    NOP
U 01, 7700,0800,5BE4,031C,2470,1802     000C   191 ;% 01    WCTRL/LOADCRAR,WB_R[TEMP7]                ;DISABLE ^P INTERRUPTS
U 02, 7700,0800,5BE4,0320,2C70,1803     0017   195 ;% 02    WCTRL/CONWRITE,WB_R[TEMP8]                ;...
U 03, 7700,0800,5BF4,0324,0070,1804     0022   199 ;% 03    PSL_R[TEMP9]                              ;SET IPL TO 1F AND NATIVE MODE
U 04, 5700,0840,0364,0328,0470,1805     002D   203 ;% 04    R[TEMP10]_RDM                             ;LOOP COUNT TO RTEMP10
U 05, 7700,D800,EF64,0301,0470,1806     0038   207 ;% 05    PL_[2]                                    ;PLATCH GETS 2
U 06, 7700,8800,8B71,0328,0470,1807     0043   211 ;% 06    Q_R[TEMP10].RL.P                          ;FORM CACHE ADDRESS
U 07, 7700,4800,03A4,03D8,4870,1808     004E   215 ;% 07    WCTRL/PC_WB,WB_Q                          ;CAUSE PREFETCH
U 08, 7700,4800,0364,1700,0470,1809     0059   219 ;% 08    IRD1TEST                                  ;LOAD INST. REGISTER
U 09, 7700,0800,5BE5,0300,0470,180A     0064   223 ;% 09    Q_R[TEMP0]                                ;LOAD Q REG WITH DATA
U 0A, 7701,8806,0090,0201,0470,180B     006F   227 ;% 0A    SIZE[LONG],WB_M[TEMP6]+Q,CCOP2            ;ADD TO MTEMP DATA SET CODES
U 0B, 6700,4800,0364,0300,0870,180C     007A   231 ;% 0B    RDM_WB,WB_PSL                             ;READ PSL TO RDM
U 0C, 7300,C800,0364,0300,0470,180D     0085   235 ;% 0C    NOP,STOP.CLOCK                            ;STOP THE CPU CLOCK
                                        0090   239
                                        0090   240 RTEMP_DATA
                                        0001   241
                                80000000 0001   242         .LONG   ^X80000000                        ;TEST DATA
                                000000C0 0005   243         .LONG   ^X000000C0
                                00000080 0009   244         .LONG   ^X00000080
                                00000000 000D   245         .LONG   ^X00000000
                                00008000 0011   246         .LONG   ^X00008000
                                00004000 0015   247         .LONG   ^X00004000
                                00004040 0019   248         .LONG   ^X00004040
                                00C00000 001D   249         .LONG   ^X00C00000                        ;DISABLES ^P INTERRUPTS
                                00400000 0021   250         .LONG   ^X00400000                        ;...
                                001F0000 0025   251         .LONG   ^X001F0000                        ;IPL = 1F NATIVE MODE
                                00000000 0029   252         .LONG   ^X00000000                        ;LOOP COUNT STORAGE
                                         002D   253
                                         002D   254 CACHE_DATA
                                         0001   255
                                0000003F 0001   256         .LONG   ^X0000003F                        ;INSTRUCTIONS
                                00000041 0005   257         .LONG   ^X00000041
                                0000000E 0009   258         .LONG   ^X0000000E
                                00000041 000D   259         .LONG   ^X00000041
                                000000C0 0011   260         .LONG   ^X000000C0
                                0000006C 0015   261         .LONG   ^X0000006C
                                000000F7 0019   262         .LONG   ^X000000F7
                                00000041 001D   263         .LONG   ^X00000041
                                00000028 0021   264         .LONG   ^X00000028
```

```
0000005C  0025   265          .LONG   ^X0000005C                      ;LOOP 10
00000058  0029   266          .LONG   ^X00000058
00000011  002D   267          .LONG   ^X00000011
00000010  0031   268          .LONG   ^X00000010
000000E5  0035   269          .LONG   ^X000000E5
00000041  0039   270          .LONG   ^X00000041
00000041  003D   271          .LONG   ^X00000041
00000078  0041   272          .LONG   ^X00000078
000000F4  0045   273          .LONG   ^X000000F4
00000048  0049   274          .LONG   ^X00000048
00000074  004D   275          .LONG   ^X00000074                      ;LOOP 20
00000041  0051   276          .LONG   ^X00000041
000000C1  0055   277          .LONG   ^X000000C1
00000041  0059   278          .LONG   ^X00000041
000000D8  005D   279          .LONG   ^X000000D8
0000000C  0061   280          .LONG   ^X0000000C
00000096  0065   281          .LONG   ^X00000096
00000041  0069   282          .LONG   ^X00000041
000000EE  006D   283          .LONG   ^X000000EE
000000F2  0071   284          .LONG   ^X000000F2
00000098  0075   285          .LONG   ^X00000098                      ;LOOP 30
0000000B  0079   286          .LONG   ^X0000000B
00000053  007D   287          .LONG   ^X00000053
0000004F  0081   288          .LONG   ^X0000004F
0000004C  0085   289          .LONG   ^X0000004C
000000A8  0089   290          .LONG   ^X000000A8
000000B0  008D   291          .LONG   ^X000000B0
00000000  0091   292          .LONG   ^X00000000
          0095   293
          0095   294 END_CPU_DATA
          01A2   295
          01A2   296
          01A2   297 END_TEST_DATA
          01A2
          01A2        ;-----------------------------------------------------------------------
          01A2   298 ENDTEST
```

```
                        0011        .SBTTL  "TEST   090     N BIT TEST 2
                        0011    300 ;*******************************************************************************
                        0011    301 ;++
                        0011    302 ;
                        0011    303 ; FUNCTIONAL DESCRIPTION:
                        0011    304 ;
                        0011    305 ;       THIS TEST VERIFIES THAT THE N BIT IN THE CCC GATE ARRAY SETS PROPERLY
                        0011    306 ;       FOR COMPATABILITY MODE INSTRUCTIONS USING CC/CCOP2.CCBR_SIGND
                        0011    307 ;       MICRO ORDER.
                        0011    308 ;
                        0011    309 ;
                        0011    310 ; TEST ALGORITHM:
                        0011    311 ;
                        0011    312 ;       1.  CREATE A TEST LOOP OF 9
                        0011    313 ;       2.  MODIFY THE MICRO CODE FOR EACH LOOP THROUGH THE TEST
                        0011    314 ;       3.  LOAD THE NEW MICRO CODE INTO THE RDM DCS
                        0011    315 ;       4.  SAVE THE LOOP COUNT
                        0011    316 ;       5.  MOVE THE LOOP COUNT TO THE RDM WDREG
                        0011    317 ;       6.  EXECUTE THE DCS PROGRAM
                        0011    318 ;               a.  DISABLE ^P INTERRUPTS
                        0011    319 ;               b.  SET THE IPL TO 1F AND MODE TO NATIVE
                        0011    320 ;               c.  MOVE THE LOOP COUNT TO THE RTEMPS
                        0011    321 ;               d.  LOAD THE PLATCH WITH 2
                        0011    322 ;               e.  ROTATE THE LOOP COUNT LEFT BY 2 TO FORM A CACHE ADDRESS
                        0011    323 ;               f.  LOAD THE PC WITH THE CACHE ADDRESS TO FORCE PREFETCH
                        0011    324 ;               g.  PERFORM IRD1TEST TO LOAD THE IR REGISTER
                        0011    325 ;               h.  LOAD THE Q REGISTER WITH TEST DATA
                        0011    326 ;               i.  SET COMPATABILITY MODE
                        0011    327 ;               j.  ADD THE Q REGISTER TO ONE OF THE MTEMPS TO CREATE THE
                        0011    328 ;                       TEST CONDITIONS WHILE PERFORMING CCOP2.
                        0011    329 ;               k.  READ THE PSL TO THE RDM
                        0011    330 ;       7.  TEST THE PSL N BIT FOR CORRECT STATE
                        0011    331 ;       8.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST PATTERNS
                        0011    332 ;
                        0011    333 ;
                        0011    334 ; TEST PATTERNS:
                        0011    335 ;
                        0011    336 ;       ITERATION       IR      Q REG. + M[TEMP*]       SIZE
                        0011    337 ;
                        0011    338 ;           1           20      00000040 + 00000040     BYTE
                        0011    339 ;           2           20      00008040 + 00000040     WORD
                        0011    340 ;           3           24      00000000 + 00000000     WORD
                        0011    341 ;           4           22      00000080 + 00000080     BYTE
                        0011    342 ;           5           20      000000C0 + 000000C0     BYTE
                        0011    343 ;           6           80      801F0000 + 00000000     LONG
                        0011    344 ;           7           24      00000040 + 00000040     BYTE
                        0011    345 ;           8           20      0000C000 + 0000C000     WORD
                        0011    346 ;           9           20      00004000 + 00004000     BYTE
                        0011    347 ;
                        0011    348 ;
                        0011    349 ; TEST ALGORITHM:
                        0011    350 ;
                        0011    351 ;       THESE TEST PATTERNS WERE GENERATED USING A LOGIC SIMULATOR.  SO IF
                        0011    352 ;       YOU ATTEMPT TO READ THE VAX SRM AND TRY AN CORRELATE THE INSTRUCTIONS
                        0011    353 ;       USED VERSUS CONDITIONS CODES IT REALLY WON'T MAKE ANY SENSE.  THE
```

```
                                0011   ;       ATTEMPT HERE IS TO TEST THE INTERNAL WORKINGS OF THE CCC GATE ARRAY.
                                0011   ;       ANY FAILUES SHOULD BE CONFINED TO THIS ARRAY OR THE INSTRUCTION
                                0011   ;       REGISTER FEEDING IT.
                                0011   ;
                                0011   ; ASSUMPTIONS:
                                0011   ;
                                0011   ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
                                0011   ;
                                0011   ; ERROR DESCRIPTION:
                                0011   ;
                                0011   ;       EXPECTED N BIT STATE
                                0011   ;       RECEIVED N BIT STATE
                                0011   ;       LOOP COUNT I
                                0011   ;
                                0011   ;--
                                0011   ;*******************************************************************************
                                0011
                                0011
                                0011
                                0011           INITIALIZE                                  ;INIT THE CPU
                                0013           LOOP            I,1,9                       ;CREATE A TEST LOOP OF 9
                                001A           LDFIELD         6$,I,B,4$,34,39,            ;MODIFY U-CODE FOR THIS LOOP
                                002B           LDFIELD         7$,I,B,5$,40,41             ;...
                                003C           LDFIELD         8$,I,B,5$,64,68             ;...
                                004D           LOADDCS         4$,,<^X09>,3                ;LOAD U-CODE INTO DCS
                                0054           SAVEINDEX       I,3$                        ;SAVE LOOP COUNT
                                0059           LOADREG         WDREG,3$,,L                 ;LOAD LOOP COUNT INTO WDREG
                                0061           ERRLOOP                                     ;ERROR LOOP POINT
                                0063           FETCH           0                           ;START DCS PROGRAM
                                0068           BRSTCLK         <^X0D>                      ;END DCS PROGRAM
                                006C           CMPREGMSK       WHREG,1$,I,2$,,L,           ;TEST THE PSL N BIT
                                0078           IFERROR         ,<CCC>,                     ;BIT INCORRECT
                                007E           ENDLOOP         I                           ;END TEST LOOP I
                                0081           SKIP                                        ;GO TO NEXT TEST
                                0088
                                0088   BEGIN_TEST_DATA
                                0088
                                0088   ;-----------------------------------------------------------------------
                                0088
                        00000000 0088   1$:     .LONG   ^X00000000                          ;RESULTS TABLE
                        00000008 008C           .LONG   ^X00000008
                        00000000 0090           .LONG   ^X00000000
                        00000000 0094           .LONG   ^X00000000
                        00000008 0098           .LONG   ^X00000008
                        00000008 009C           .LONG   ^X00000008
                        00000008 00A0           .LONG   ^X00000008
                        00000008 00A4           .LONG   ^X00000008
                        00000000 00A8           .LONG   ^X00000000
                        00000008 00AC   2$:     .LONG   ^X00000008                          ;MASK FOR CMPREGMSK PSEUDO OP
                        00000000 00B0   3$:     .LONG   ^X00000000                          ;LOOP COUNT STORAGE
                                00B4   4$:
U 09, 7700,0800,5BE5,0300,0470,180A 00B4   ;% 09   Q_R[TEMP0]                                 ;U-CODE TO MODIFY EACH
U 0A, 7700,0800,5BF4,0328,0070,180B 00BF   ;% 0A   PSL_R[TEMP10]                              ;LOOP
                                00CA   5$:
U 0B, 7701,4800,0090,0001,0470,180C 00CA   ;% 0B   SIZE[BYTE],WB_M[TEMP0]+Q,CCOP2             ;...
```

```
                    04 03 02 01 00  00D5   416 6$:     .BYTE   00,01,02,03,04                  ;RSRC LDFIELD TABLE
                       05 06 00 0A  00DA   417         .BYTE   10,00,06,05
                    00 00 01 01 00  00DE   418 7$:     .BYTE   00,01,01,00,00                  ;SIZE LDFIELD TABLE
                       00 01 00 02  00E3   419         .BYTE   02,00,01,00
                    04 03 02 00 00  00E7   420 8$:     .BYTE   00,00,02,03,04                  ;MSRC LDFIELD TABLE
                       05 06 00 02  00EC   421         .BYTE   02,00,06,05
                                    00F0   422
                                    00F0   423
                                    00F0   424 BEGIN_CPU_DATA
                                    0002   425
                                    0002   426 DCS_DATA
                                    0001   427
U 00. 7700,C800,0364,0300,0470,1801 0001   428 ;% 00   NOP
U 01. 7700,0800,5BE4,031C,2470,1802 000C   432 ;% 01   WCTRL/LOADCRAR,WB_R[TEMP7]               ;DISABLE ^P INTERRPUTS
U 02. 7700,0800,5BE4,0320,2C70,1803 0017   436 ;% 02   WCTRL/CONWRITE,WB_R[TEMP8]              ;...
U 03. 7700,0800,5BF4,0324,0070,1804 0022   440 ;% 03   PSL_R[TEMP9]                           ;SET IPL = 1F AND NATIVE MODE
U 04. 5700,4840,0364,032C,0470,1805 002D   444 ;% 04   R[TEMP11]_RDM                          ;LOOP COUNT TO RTEMP11
U 05. 7700,D800,EF64,0301,0470,1806 0038   448 ;% 05   PL_[2]                                ;PLATCH GETS 2
U 06. 7700,C800,8B71,032C,0470,1807 0043   452 ;% 06   Q_R[TEMP11].RL.P                      ;FORM CACHE ADDRESS
U 07. 7700,4800,03A4,03D8,4870,1808 004E   456 ;% 07   WCTRL/PC_WB,WB_Q                       ;CAUSE PREFETCH
U 08. 7700,4800,0364,1700,0470,1809 0059   460 ;% 08   IRD1TEST                              ;LOAD INST. REGISTER
U 09. 7700,0800,5BE5,0300,0470,180A 0064   464 ;% 09   Q_R[TEMP0]                            ;Q GETS TEST DATA
U 0A. 7700,0800,5BF4,0328,0070,180B 006F   468 ;% 0A   PSL_R[TEMP10]                          ;SET COMPATABILITY MODE
U 0B. 7701,4800,0090,0001,0470,180C 007A   472 ;% 0B   SIZE[BYTE],WB_M[TEMP0]+Q,CCOP2        ;SET CONDITION CODES
U 0C. 6700,C800,0364,0300,0870,180D 0085   476 ;% 0C   RDM_WB,WB_PSL                         ;READ PSL TO RDM
U 0D. 7300,C800,0364,0300,0470,180E 0090   480 ;% 0D   NOP,STOP.CLOCK                       ;STOP THE CPU CLOCK
                                    009B   484
                                    009B   485 RTEMP_DATA
                                    0001   486
                    00000040        0001   487         .LONG   ^X00000040                      ;TEST DATA
                    00008040        0005   488         .LONG   ^X00008040
                    00000000        0009   489         .LONG   ^X00000000
                    00000080        000D   490         .LONG   ^X00000080
                    000000C0        0011   491         .LONG   ^X000000C0
                    00004000        0015   492         .LONG   ^X00004000
                    0000C000        0019   493         .LONG   ^X0000C000
                    00C00000        001D   494         .LONG   ^X00C00000                      ;DISABLE ^P INTERRUPTS
                    00400000        0021   495         .LONG   ^X00400000                      ;...
                    001F0000        0025   496         .LONG   ^X001F0000                      ;IPL = 1F NATIVE MODE
                    801F0000        0029   497         .LONG   ^X801F0000                      ;IPL = 1F COMPATABILITY MODE
                    00000000        002D   498         .LONG   ^X00000000                      ;LOOP COUNT STORAGE
                                    0031   499
                                    0031   500 CACHE_DATA
                                    0001   501
                    00000020        0001   502         .LONG   ^X00000020                      ;INSTRUCTION TABLE
                    00000020        0005   503         .LONG   ^X00000020
                    00000024        0009   504         .LONG   ^X00000024
                    00000022        000D   505         .LONG   ^X00000022
                    00000020        0011   506         .LONG   ^X00000020
                    00000080        0015   507         .LONG   ^X00000080
                    00000024        0019   508         .LONG   ^X00000024
                    00000020        001D   509         .LONG   ^X00000020
                    00000020        0021   510         .LONG   ^X00000020
                    00000000        0025   511         .LONG   ^X00000000
                                    0029   512
```

```
                        0029    513 END_CPU_DATA
                        00F0    514
                        00F0    515
                        00F0    516 END_TEST_DATA
                        00F0
                        00F0        ;-----------------------------------------------------------------------
                        00F0    517 ENDTEST
```

```
                0011        .SBTTL  "TEST   091     N BIT TEST 3
                0011    519 ;*****************************************************************************
                0011    520 ;++
                0011    521 ;
                0011    522 ; FUNCTIONAL DESCRIPTION:
                0011    523 ;
                0011    524 ;       THIS TEST VERIFIES THAT THE N BIT IN THE CCC GATE ARRAY SETS PROPERLY
                0011    525 ;       FOR NATIVE MODE INSTRUCTIONS USING CC/CCOP1.CCBR_SIGND MICRO ORDER.
                0011    526 ;
                0011    527 ;
                0011    528 ; TEST ALGORITHM:
                0011    529 ;
                0011    530 ;       1.  CREATE A TEST LOOP OF 45
                0011    531 ;       2.  MODIFY THE MICRO CODE FOR EACH LOOP THROUGH THE TEST
                0011    532 ;       3.  LOAD THE NEW MICRO CODE INTO THE RDM DCS
                0011    533 ;       4.  SAVE THE LOOP COUNT
                0011    534 ;       5.  MOVE THE LOOP COUNT TO THE RDM WDREG
                0011    535 ;       6.  EXECUTE THE DCS PROGRAM
                0011    536 ;               a.  DISABLE ^P INTERRUPTS
                0011    537 ;               b.  SET THE IPL TO 1F AND MODE TO NATIVE
                0011    538 ;               c.  MOVE THE LOOP COUNT TO THE RTEMPS
                0011    539 ;               d.  LOAD THE PLATCH WITH 2
                0011    540 ;               e.  ROTATE THE LOOP COUNT LEFT BY 2 TO FORM A CACHE ADDRESS
                0011    541 ;               f.  LOAD THE PC WITH THE CACHE ADDRESS TO FORCE PREFETCH
                0011    542 ;               g.  PERFORM IRD1TEST TO LOAD THE IR REGISTER
                0011    543 ;               h.  LOAD THE Q REGISTER WITH TEST DATA
                0011    544 ;               i.  ADD THE Q REGISTER TO ONE OF THE RTEMPS TO CREATE THE
                0011    545 ;                       TEST CONDITIONS WHILE PERFORMING CCOP1.
                0011    546 ;               j.  READ THE PSL TO THE RDM
                0011    547 ;       7.  TEST THE PSL N BIT FOR CORRECT STATE
                0011    548 ;       8.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST PATTERNS
                0011    549 ;
                0011    550 ;
                0011    551 ; TEST PATTERNS:
                0011    552 ;
                0011    553 ;       ITERATION       IR      Q REG. + R[TEMP*]       SIZE
                0011    554 ;
                0011    555 ;           1           F0      00000040 + 00000040     BYTE
                0011    556 ;           2           6C      00008000 + 00000000     LONG
                0011    557 ;           3           CF      80000000 + 80000000     LONG
                0011    558 ;           4           31      0000C000 + 0000C000     WORD
                0011    559 ;           5           90      00004000 + 00004000     WORD
                0011    560 ;           6           EE      80000080 + 80000080     WORD
                0011    561 ;           7           FB      80000000 + 00000000     LONG
                0011    562 ;           8           59      00008000 + 00000000     WORD
                0011    563 ;           9           1B      00000080 + 00000000     BYTE
                0011    564 ;           A           30      00000000 + 00000000     WORD
                0011    565 ;           B           EE      00008080 + 00008080     WORD
                0011    566 ;           C           74      00000080 + 00000000     BYTE
                0011    567 ;           D           E2      80000000 + 00000000     LONG
                0011    568 ;           E           04      80008080 + 80000080     WORD
                0011    569 ;           F           D0      00000040 + 00000040     BYTE
                0011    570 ;          10           EE      80008000 + 80008000     BYTE
                0011    571 ;          11           72      00008000 + 00000000     LONG
                0011    572 ;          12           EB      00000080 + 00000000     BYTE
```

```
0011    573 ;          13          30      00000000 + 00000000     BYTE
0011    574 ;          14          30      00000000 + 00000000     LONG
0011    575 ;          15          5C      00000080 + 00000000     BYTE
0011    576 ;          16          B3      00000040 + 00000040     BYTE
0011    577 ;          17          30      000000C0 + 000000C0     BYTE
0011    578 ;          18          A6      00000080 + 00000000     BYTE
0011    579 ;          19          9A      00008000 + 00000000     WORD
0011    580 ;          1A          D1      00000040 + 00000040     BYTE
0011    581 ;          1B          8E      00000080 + 00000000     BYTE
0011    582 ;          1C          D3      00000040 + 00000040     BYTE
0011    583 ;          1D          20      00008080 + 00008080     LONG
0011    584 ;          1E          30      80000000 + 80000000     LONG
0011    585 ;          1F          8F      40000000 + 40000000     LONG
0011    586 ;          20          F7      00000080 + 00000000     BYTE
0011    587 ;          21          4F      00000040 + 00000040     BYTE
0011    588 ;          22          B1      00000040 + 00000040     BYTE
0011    589 ;          23          C9      00000080 + 00000000     BYTE
0011    590 ;          24          4A      80000000 + 00000000     LONG
0011    591 ;          25          0E      00000040 + 00000040     BYTE
0011    592 ;          26          32      80000000 + 00000000     LONG
0011    593 ;          27          58      80000000 + 00000000     LONG
0011    594 ;          28          C6      80000000 + 00000000     LONG
0011    595 ;          29          8D      80000000 + 80000000     LONG
0011    596 ;          2A          60      00000040 + 00000000     LONG
0011    597 ;          2B          10      00000080 + 00000080     BYTE
0011    598 ;          2C          48      00000000 + 00000000     LONG
0011    599 ;          2D          8E      00000080 + 00000080     BYTE
0011    600 ;
0011    601 ;
0011    602 ; TEST ALGORITHM:
0011    603 ;
0011    604 ;       THESE TEST PATTERNS WERE GENERATED USING A LOGIC SIMULATOR.  SO IF
0011    605 ;       YOU ATTEMPT TO READ THE VAX SRM AND TRY AN CORRELATE THE INSTRUCTIONS
0011    606 ;       USED VERSUS CONDITIONS CODES IT REALLY WON'T MAKE ANY SENSE.  THE
0011    607 ;       ATTEMPT HERE IS TO TEST THE INTERNAL WORKINGS OF THE CCC GATE ARRAY.
0011    608 ;       ANY FAILUES SHOULD BE CONFINED TO THIS ARRAY OR THE INSTRUCTION
0011    609 ;       REGISTER FEEDING IT.
0011    610 ;
0011    611 ; ASSUMPTIONS:
0011    612 ;
0011    613 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
0011    614 ;
0011    615 ; ERROR DESCRIPTION:
0011    616 ;
0011    617 ;       EXPECTED N BIT STATE
0011    618 ;       RECEIVED N BIT STATE
0011    619 ;       LOOP COUNT I
0011    620 ;
0011    621 ;--
0011    622 ;*******************************************************************************
0011    623
0011    624
0011    625         INITIALIZE                                  ;INT THE CPU
0013    626         LOOP            I,1,45                      ;CREATE A TEST LOOP OF 45
001A    627         LDFIELD         6$,I,B,4$,34,39,            ;MODIFY U-CODE FOR THIS LOOP
```

```
                002B          LDFIELD        7$,I,B,5$,40,41,          ;...
                003C          LDFIELD        8$,I,B,5$,34,39,          ;...
                004D          LOADDCS        4$,,09,2                  ;LOAD U-CODE INTO DCS
                0054          SAVEINDEX      I,3$                      ;SAVE LOOP COUNT
                0059          LOADREG        WDREG,3$,,L               ;PUT LOOP COUNT IN WDREG
                0061          ERRLOOP                                  ;ERROR LOOP POINT
                0063          FETCH          0                         ;START DCS PROGRAM
                0068          BRSTCLK        <^X0C>                    ;END DCS PROGRAM
                006C          CMPREGMSK      WHREG,1$,I,2$,,L,         ;TEST FOR N BIT
                0078          IFERROR        ,<CCC>                    ;BIT INCORRECT
                007E          ENDLOOP        I                         ;END TEST LOOP
                0081          SKIP                                     ;GO TO NEXT TEST
                0088
                0088
                0088    BEGIN_TEST_DATA
                0088
                0088    ;-----------------------------------------------------------------------
                0088
        00000000 0088    1$:     .LONG   ^X00000000                      ;RESULTS TABLE
        00000008 008C            .LONG   ^X00000008
        00000008 0090            .LONG   ^X00000008
        00000000 0094            .LONG   ^X00000000
        00000008 0098            .LONG   ^X00000008
        00000000 009C            .LONG   ^X00000000
        00000008 00A0            .LONG   ^X00000008
        00000008 00A4            .LONG   ^X00000008
        00000000 00A8            .LONG   ^X00000000
        00000008 00AC            .LONG   ^X00000008                      ;LOOP 10
        00000008 00B0            .LONG   ^X00000008
        00000000 00B4            .LONG   ^X00000000
        00000008 00B8            .LONG   ^X00000008
        00000008 00BC            .LONG   ^X00000008
        00000008 00C0            .LONG   ^X00000008
        00000000 00C4            .LONG   ^X00000000
        00000008 00C8            .LONG   ^X00000008
        00000000 00CC            .LONG   ^X00000000
        00000008 00D0            .LONG   ^X00000008
        00000008 00D4            .LONG   ^X00000008                      ;LOOP 20
        00000008 00D8            .LONG   ^X00000008
        00000008 00DC            .LONG   ^X00000008
        00000000 00E0            .LONG   ^X00000000
        00000008 00E4            .LONG   ^X00000008
        00000008 00E8            .LONG   ^X00000008
        00000000 00EC            .LONG   ^X00000000
        00000008 00F0            .LONG   ^X00000008
        00000008 00F4            .LONG   ^X00000008
        00000000 00F8            .LONG   ^X00000000
        00000000 00FC            .LONG   ^X00000000                      ;LOOP 30
        00000000 0100            .LONG   ^X00000000
        00000008 0104            .LONG   ^X00000008
        00000000 0108            .LONG   ^X00000000
        00000000 010C            .LONG   ^X00000000
        00000008 0110            .LONG   ^X00000008
        00000008 0114            .LONG   ^X00000008
        00000000 0118            .LONG   ^X00000000
```

```
                        00000008  011C   681          .LONG   ^X00000008
                        00000008  0120   682          .LONG   ^X00000008
                        00000008  0124   683          .LONG   ^X00000008                      ;LOOP 40
                        00000000  0128   684          .LONG   ^X00000000
                        00000008  012C   685          .LONG   ^X00000008
                        00000000  0130   686          .LONG   ^X00000000
                        00000000  0134   687          .LONG   ^X00000000
                        00000000  0138   688          .LONG   ^X00000000
                        00000008  013C   689 2$:      .LONG   ^X00000008                      ;MASK FOR CMPREGMSK PSEUDO OP
                        00000000  0140   690 3$:      .LONG   ^X00000000                      ;LOOP COUNT STORAGE
                                  0144   691 4$:
U 09, 7700,0800,5BE5,0300,0470,180A       0144   692 ;% 09    Q_R[TEMP0]                      ;U-CODE MODIFIED EACH
                                  014F   696 5$:                                             ;LOOP
U 0A, 7701,C800,0390,0000,8470,180B       014F   697 ;% 0A    SIZE[BYTE],ALU/A+B+CI,MUX/R.Q,RSRC/TEMP0,CCOP1
      07 06 01 02 05 04 03 02 01 00       015A   701 6$:      .BYTE   00,01,02,03,04,05,02,01,06,07   ;RSRC LDFIELD TABLE
      07 07 06 01 0C 00 09 02 06 08       0164   702          .BYTE   08,06,02,09,00,12,01,06,07,07
      02 08 00 06 00 01 06 0A 00 06       016E   703          .BYTE   06,00,10,06,01,00,06,00,08,02
      02 02 02 00 02 06 00 00 06 0B       0178   704          .BYTE   11,06,00,00,06,02,00,02,02,02
                  06 07 06 01 02          0182   705          .BYTE   02,01,06,07,06
                                  0187   706
      01 00 01 02 01 01 01 02 02 00       0187   707 7$:      .BYTE   00,02,02,01,01,01,02,01,00,01   ;SIZE LDFIELD TABLE
      02 00 00 02 00 00 01 02 00 01       0191   708          .BYTE   01,00,02,01,00,00,02,00,00,02
      02 02 00 00 00 01 00 00 00 00       019B   709          .BYTE   00,00,00,00,01,00,00,00,02,02
      02 02 02 00 02 00 00 00 00 02       01A5   710          .BYTE   02,00,00,00,00,02,00,02,02,02
                  00 02 00 02 02          01AF   711          .BYTE   02,02,00,02,00
                                  01B4   712
      07 07 07 07 05 04 03 02 07 00       01B4   713 8$:      .BYTE   00,07,02,03,04,05,07,07,07,07   ;RSRC LDFIELD TABLE
      07 07 07 07 0C 00 05 07 07 08       01BE   714          .BYTE   08,07,07,05,00,12,07,07,07,07
      02 08 00 07 00 07 07 0A 00 07       01C8   715          .BYTE   07,00,10,07,07,00,07,00,08,02
      07 07 07 00 07 07 00 00 07 0B       01D2   716          .BYTE   11,07,00,00,07,07,00,07,07,07
                  06 07 06 07 02          01DC   717          .BYTE   02,07,06,07,06
                                  01E1   718
                                  01E1   719
                                  01E1   720 BEGIN_CPU_DATA
                                  0002   721
                                  0002   722 DCS_DATA
                                  0001   723
U 00, 7700,C800,0364,0300,0470,1801       0001   724 ;% 00    NOP
U 01, 7700,0800,5BE4,0334,2470,1802       000C   728 ;% 01    WCTRL/LOADCRAR,WB_R[TEMP13]     ;DISABLE ^P INTERRUPTS
U 02, 7700,0800,5BE4,0338,2C70,1803       0017   732 ;% 02    WCTRL/CONWRITE,WB_R[MM.TEMP5]   ;...
U 03, 7700,0800,5BF4,033C,0070,1804       0022   736 ;% 03    PSL_R[MM.TEMP1]                 ;SET IPL = 1F MODE NATIVE
U 04, 5700,4840,0364,0340,0470,1805       002D   740 ;% 04    R[R0]_RDM                       ;LOOP COUNT TO RTEMPS
U 05, 7700,D800,EF64,0301,0470,1806       0038   744 ;% 05    PL_[2]                          ;PLATCH GETS 2
U 06, 7700,C800,8B71,0340,0470,1807       0043   748 ;% 06    Q_R[R0].RL.P                    ;FORM CACHE ADDRESS
U 07, 7700,4800,03A4,03D8,4870,1808       004E   752 ;% 07    WCTRL/PC_WB,WB_Q                ;FORCE PREFETCH
U 08, 7700,4800,0364,1700,0470,1809       0059   756 ;% 08    IRD1TEST                        ;LOAD INST. REGISTER
U 09, 7700,0800,5BE5,0300,0470,180A       0064   760 ;% 09    Q_R[TEMP0]                      ;Q REGISTER GETS TEST DATA
U 0A, 7701,C800,0390,0000,8470,180B       006F   764 ;% 0A    SIZE[BYTE],ALU/A+B+CI,MUX/R.Q,RSRC/TEMP0,CCOP1;SET COND CODES
U 0B, 6700,4800,0364,0300,0870,180C       007A   768 ;% 0B    RDM_WB,WB_PSL                   ;READ PSL TO RDM
U 0C, 7300,C800,0364,0300,0470,180D       0085   772 ;% 0C    NOP,STOP.CLOCK                  ;STOP THE CPU CLOCK
                                  0090   776
                                  0090   777 RTEMP_DATA
                                  0001   778
                        00000040  0001   779          .LONG   ^X00000040                      ;TEST PATTERN TABLE
                        00008000  0005   780          .LONG   ^X00008000
```

```
80000000  0009   781          .LONG   ^X80000000
0000C000  000D   782          .LONG   ^X0000C000
00004000  0011   783          .LONG   ^X00004000
80000080  0015   784          .LONG   ^X80000080
00000080  0019   785          .LONG   ^X00000080
00000000  001D   786          .LONG   ^X00000000
00008080  0021   787          .LONG   ^X00008080
80008080  0025   788          .LONG   ^X80008080
000000C0  0029   789          .LONG   ^X000000C0
40000000  002D   790          .LONG   ^X40000000
80008000  0031   791          .LONG   ^X80008000
00C00000  0035   792          .LONG   ^X00C00000                      ;DISABLE ^P INTERRUPTS
00400000  0039   793          .LONG   ^X00400000                      ;...
001F0000  003D   794          .LONG   ^X001F0000                      ;IPL = 1F NATIVE MODE
00000000  0041   795          .LONG   ^X00000000                      ;LOOP COUNT STORAGE
          0045   796
          0045   797 CACHE_DATA
          0001   798
000000F0  0001   799          .LONG   ^X000000F0                      ;INSTRUCTION TABLE
0000006C  0005   800          .LONG   ^X0000006C
000000CF  0009   801          .LONG   ^X000000CF
00000031  000D   802          .LONG   ^X00000031
00000090  0011   803          .LONG   ^X00000090
000000EE  0015   804          .LONG   ^X000000EE
000000FB  0019   805          .LONG   ^X000000FB
00000059  001D   806          .LONG   ^X00000059
0000001B  0021   807          .LONG   ^X0000001B
00000030  0025   808          .LONG   ^X00000030                      ;LOOP 10
000000EE  0029   809          .LONG   ^X000000EE
00000074  002D   810          .LONG   ^X00000074
000000E2  0031   811          .LONG   ^X000000E2
00000004  0035   812          .LONG   ^X00000004
000000D0  0039   813          .LONG   ^X000000D0
000000EE  003D   814          .LONG   ^X000000EE
00000072  0041   815          .LONG   ^X00000072
000000EB  0045   816          .LONG   ^X000000EB
00000030  0049   817          .LONG   ^X00000030
00000030  004D   818          .LONG   ^X00000030                      ;LOOP 20
0000005C  0051   819          .LONG   ^X0000005C
000000B3  0055   820          .LONG   ^X000000B3
00000030  0059   821          .LONG   ^X00000030
000000A6  005D   822          .LONG   ^X000000A6
0000009A  0061   823          .LONG   ^X0000009A
000000D1  0065   824          .LONG   ^X000000D1
0000008E  0069   825          .LONG   ^X0000008E
000000D3  006D   826          .LONG   ^X000000D3
00000020  0071   827          .LONG   ^X00000020
00000030  0075   828          .LONG   ^X00000030                      ;LOOP 30
0000008F  0079   829          .LONG   ^X0000008F
000000F7  007D   830          .LONG   ^X000000F7
0000004F  0081   831          .LONG   ^X0000004F
000000B1  0085   832          .LONG   ^X000000B1
000000C9  0089   833          .LONG   ^X000000C9
0000004A  008D   834          .LONG   ^X0000004A
0000000E  0091   835          .LONG   ^X0000000E
```

```
00000032  0095   836          .LONG   ^X00000032
00000058  0099   837          .LONG   ^X00000058
000000C6  009D   838          .LONG   ^X000000C6                    ;LOOP 40
0000008D  00A1   839          .LONG   ^X0000008D
00000060  00A5   840          .LONG   ^X00000060
00000010  00A9   841          .LONG   ^X00000010
00000048  00AD   842          .LONG   ^X00000048
0000008E  00B1   843          .LONG   ^X0000008E
          00B5   844
          00B5   845 END_CPU_DATA
          01E1   846
          01E1   847
          01E1   848 END_TEST_DATA
          01E1
          01E1       ;-------------------------------------------------------------------------
          01E1   849 ENDTEST
```

```
                        0011        .SBTTL  "TEST   092     N BIT TEST 4
                        0011    851 ;*******************************************************************************
                        0011    852 ;++
                        0011    853 ;
                        0011    854 ; FUNCTIONAL DESCRIPTION:
                        0011    855 ;
                        0011    856 ;       THIS TEST VERIFIES THAT THE N BIT IN THE CCC GATE ARRAY SETS PROPERLY
                        0011    857 ;       FOR COMPATABILITY MODE INSTRUCTIONS USING CC/CCOP1.CCBR_SIGND
                        0011    858 ;       MICRO ORDER.
                        0011    859 ;
                        0011    860 ;
                        0011    861 ; TEST ALGORITHM:
                        0011    862 ;
                        0011    863 ;       1.  CREATE A TEST LOOP OF 4
                        0011    864 ;       2.  MODIFY THE MICRO CODE FOR EACH LOOP THROUGH THE TEST
                        0011    865 ;       3.  LOAD THE NEW MICRO CODE INTO THE RDM DCS
                        0011    866 ;       4.  SAVE THE LOOP COUNT
                        0011    867 ;       5.  MOVE THE LOOP COUNT TO THE RDM WDREG
                        0011    868 ;       6.  EXECUTE THE DCS PROGRAM
                        0011    869 ;               a.  SET THE IPL TO 1F AND MODE TO NATIVE
                        0011    870 ;               b.  DISABLE ^P INTERRUPTS
                        0011    871 ;               c.  MOVE THE LOOP COUNT TO THE RTEMPS
                        0011    872 ;               d.  LOAD THE PLATCH WITH 2
                        0011    873 ;               e.  ROTATE THE LOOP COUNT LEFT BY 2 TO FORM A CACHE ADDRESS
                        0011    874 ;               f.  LOAD THE PC WITH THE CACHE ADDRESS TO FORCE PREFETCH
                        0011    875 ;               g.  PERFORM IRD1TEST TO LOAD THE IR REGISTER
                        0011    876 ;               h.  LOAD THE LONLIT REGISTER WITH TEST DATA
                        0011    877 ;               i.  SET IPL TO 1F AND MODE TO COMPATABILBITY
                        0011    878 ;               j.  SET THE CONDITION CODES WITH THE TEST PATTERN ON THE WBUS
                        0011    879 ;               k.  READ THE PSL TO THE RDM
                        0011    880 ;       7.  TEST THE PSL N BIT FOR CORRECT STATE
                        0011    881 ;       8.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST PATTERNS
                        0011    882 ;
                        0011    883 ;
                        0011    884 ; TEST PATTERNS:
                        0011    885 ;
                        0011    886 ;       ITERATION       IR      TEST PATTERN    SIZE
                        0011    887 ;
                        0011    888 ;           1           06        00000000      BYTE
                        0011    889 ;           2           00        00008000      WORD
                        0011    890 ;           3           00        00000000      WORD
                        0011    891 ;           4           00        00000080      BYTE
                        0011    892 ;
                        0011    893 ;
                        0011    894 ; TEST ALGORITHM:
                        0011    895 ;
                        0011    896 ;       THESE TEST PATTERNS WERE GENERATED USING A LOGIC SIMULATOR.  SO IF
                        0011    897 ;       YOU ATTEMPT TO READ THE VAX SRM AND TRY AN CORRELATE THE INSTRUCTIONS
                        0011    898 ;       USED VERSUS CONDITIONS CODES IT REALLY WON'T MAKE ANY SENSE.  THE
                        0011    899 ;       ATTEMPT HERE IS TO TEST THE INTERNAL WORKINGS OF THE CCC GATE ARRAY.
                        0011    900 ;       ANY FAILUES SHOULD BE CONFINED TO THIS ARRAY OR THE INSTRUCTION
                        0011    901 ;       REGISTER FEEDING IT.
                        0011    902 ;
                        0011    903 ; ASSUMPTIONS:
                        0011    904 ;
```

```
                                0011   905 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
                                0011   906 ;
                                0011   907 ; ERROR DESCRIPTION:
                                0011   908 ;
                                0011   909 ;       EXPECTED N BIT STATE
                                0011   910 ;       RECEIVED N BIT STATE
                                0011   911 ;       LOOP COUNT I
                                0011   912 ;
                                0011   913 ;--
                                0011   914 ;*******************************************************************************
                                0011   915
                                0011   916
                                0011   917         INITIALIZE                              ;INIT THE CPU
                                0013   918         LOOP            I,1,4                   ;CREATE A TEST LOOP OF 4
                                001A   919         LDFIELD         1$,I,L,3$,31,62,LON     ;MODIFY U-CODE FOR THIS LOOP
                                002B   920         LDFIELD         2$,I,B,4$,40,41,        ;...
                                003C   921         LOADDCS         3$,,09,3                ;LOAD NEW U-CODE INTO DCS
                                0043   922         SAVEINDEX       I,5$                    ;SAVE LOOP COUNT
                                0048   923         LOADREG         WDREG,5$,,L             ;MOVE LOOP COUNT TO WDREG
                                0050   924         ERRLOOP                                 ;ERROR LOOP POINT
                                0052   925         FETCH           0                       ;START DCS PROGRAM
                                0057   926         BRSTCLK         <^X0D>                  ;END DCS PROGRAM
                                005B   927         CMPREGMSK       WHREG,6$,I,7$,,L,       ;TEST N BIT
                                0067   928         IFERROR         ,<CCC>,                 ;BIT INCORRECT
                                006D   929         ENDLOOP         I                       ;END TEST LOOP
                                0070   930         SKIP                                    ;GO TO NEXT TEST
                                0077   931
                                0077   932
                                0077   933 BEGIN_TEST_DATA
                                0077
                                0077         ;-------------------------------------------------------------------------
                                0077   934
                    00000000    0077   935 1$:      .LONG   ^X00000000                      ;TABLE OF LONLIT DATA
                    00008000    007B   936         .LONG   ^X00008000
                    00000000    007F   937         .LONG   ^X00000000
                    00000080    0083   938         .LONG   ^X00000080
                 00 01 01 00    0087   939 2$:      .BYTE   00,01,01,00                     ;TABLE OF SIZE DATA
                                008B   940 3$:
U 09, 7700,F800,7FFF,FFFF,8470,180A 008B   941 ;% 09   LONLIT_[00000000]                   ;DATA FOR WBUS
U 0A, 7700,0800,5BF4,030C,0070,180B 0096   945 ;% 0A   PSL_R[TEMP3]                        ;SETS COMPATABILITY MODE
                                00A1   949 4$:
U 0B, 7701,0800,5BE4,00D4,8470,180C 00A1   950 ;% 0B   SIZE[BYTE],WB_R[LONLIT],CCOP1      ;SETS CONDITION CODES
                    00000000    00AC   954 5$:      .LONG   ^X00000000                      ;LOOP COUNT STORAGE
                    00000000    00B0   955 6$:      .LONG   ^X00000000                      ;RESULTS TABLE
                    00000008    00B4   956         .LONG   ^X00000008
                    00000000    00B8   957         .LONG   ^X00000000
                    00000008    00BC   958         .LONG   ^X00000008
                    00000008    00C0   959 7$:      .LONG   ^X00000008                      ;MASK FOR CMPREGMSK PSEUDO OP
                                00C4   960
                                00C4   961
                                00C4   962 BEGIN_CPU_DATA
                                0002   963
                                0002   964 DCS_DATA
                                0001   965
U 00, 7700,C800,0364,0300,0470,1801 0001   966 ;% 00   NOP
```

```
U 01. 7700.0800.5BF4.0300.0070.1802  000C   970 ;% 01   PSL_R[TEMP0]                          ;SET NATIVE MODE
U 02. 7700.8800.5BE4.0304.2470.1803  0017   974 ;% 02   WCTRL/LOADCRAR,WB_R[TEMP1]           ;DISABLE ^P INTERRUPT
U 03. 7700.8800.5BE4.0308.2C70.1804  0022   978 ;% 03   WCTRL/CONWRITE,WB_R[TEMP2]          ;...
U 04. 5700.4840.0364.0310.0470.1805  002D   982 ;% 04   R[TEMP4]_RDM                         ;LOAD LOOP COUNT
U 05. 7700.D800.EF64.0301.0470.1806  0038   986 ;% 05   PL_[2]                              ;PLATCH GETS 2
U 06. 7700.C800.8B71.0310.0470.1807  0043   990 ;% 06   Q_R[TEMP4].RL.P                     ;FORM CACHE ADDRESS
U 07. 7700.4800.03A4.03D8.4870.1808  004E   994 ;% 07   WCTRL/PC_WB,WB_Q                    ;PREFETCH INSTRUCTION
U 08. 7700.4800.0364.1700.0470.1809  0059   998 ;% 08   IRD1TEST                            ;LOAD INSTRUCTION REGISTER
U 09. 7700.F800.7FFF.FFFF.8470.180A  0064  1002 ;% 09   LONLIT_[00000000]                   ;PATTERN FOR WBUS
U 0A. 7700.0800.5BF4.030C.0070.180B  006F  1006 ;% 0A   PSL_R[TEMP3]                        ;SET COMPATIBILITY MODE
U 0B. 7701.0800.5BE4.00D4.8470.180C  007A  1010 ;% 0B   SIZE[BYTE],WB_R[LONLIT],CCOP1      ;SET CONDITION CODES
U 0C. 6700.C800.0364.0300.0870.180D  0085  1014 ;% 0C   RDM_WB,WB_PSL                       ;PSL TO RDM
U 0D. 7300.C800.0364.0300.0470.180E  0090  1018 ;% 0D   NOP,STOP.CLOCK                     ;STOP THE CPU CLOCK
                                     009B  1022
                                     009B  1023 RTEMP_DATA
                                     0001  1024
                          001F0000   0001  1025          .LONG   ^X001F0000                ;SET NATIVE MODE
                          00C00000   0005  1026          .LONG   ^X00C00000                ;DISABLE ^P INTERRUPT
                          00400000   0009  1027          .LONG   ^X00400000                ;...
                          801F0000   000D  1028          .LONG   ^X801F0000                ;SET COMPATABILITY MODE
                          00000000   0011  1029          .LONG   ^X00000000                ;LOOP COUNT STORAGE
                                     0015  1030
                                     0015  1031 CACHE_DATA
                                     0001  1032
                          00000006   0001  1033          .LONG   ^X00000006                ;TEST INSTRUCTIONS TABLE
                          00000000   0005  1034          .LONG   ^X00000000
                          00000000   0009  1035          .LONG   ^X00000000
                          00000000   000D  1036          .LONG   ^X00000000
                          00000000   0011  1037          .LONG   ^X00000000
                                     0015  1038
                                     0015  1039 END_CPU_DATA
                                     00C4  1040
                                     00C4  1041
                                     00C4  1042 END_TEST_DATA
                                     00C4
                                     00C4          ;-----------------------------------------------------------------------------
                                     00C4  1043 ENDTEST
```

```
                        0011        .SBTTL  "TEST   093     Z BIT TEST 1
                        0011  1045 ;*******************************************************************************
                        0011  1046 ;++
                        0011  1047 ;
                        0011  1048 ; FUNCTIONAL DESCRIPTION:
                        0011  1049 ;
                        0011  1050 ;       THIS TEST VERIFIES THAT THE Z BIT IN THE CCC GATE ARRAY SETS PROPERLY
                        0011  1051 ;       FOR NATIVE MODE INSTRUCTIONS USING CC/CCOP2.CCBR_SIGND MICRO ORDER.
                        0011  1052 ;
                        0011  1053 ;
                        0011  1054 ; TEST ALGORITHM:
                        0011  1055 ;
                        0011  1056 ;       1.  CREATE A TEST LOOP OF 27
                        0011  1057 ;       2.  MODIFY THE MICRO CODE FOR EACH LOOP THROUGH THE TEST
                        0011  1058 ;       3.  LOAD THE NEW MICRO CODE INTO THE RDM DCS
                        0011  1059 ;       4.  SAVE THE LOOP COUNT
                        0011  1060 ;       5.  MOVE THE LOOP COUNT TO THE RDM WDREG
                        0011  1061 ;       6.  EXECUTE THE DCS PROGRAM
                        0011  1062 ;               a.  DISABLE ^P INTERRUPTS
                        0011  1063 ;               b.  DISABLE THE FPA (INCASE IT'S PRESENT)
                        0011  1064 ;               c.  SET THE IPL TO 1F AND MODE TO NATIVE
                        0011  1065 ;               d.  MOVE THE LOOP COUNT TO THE RTEMPS
                        0011  1066 ;               e.  LOAD THE PLATCH WITH 2
                        0011  1067 ;               f.  ROTATE THE LOOP COUNT LEFT BY 2 TO FORM A CACHE ADDRESS
                        0011  1068 ;               g.  LOAD THE PC WITH THE CACHE ADDRESS TO FORCE PREFETCH
                        0011  1069 ;               h.  PERFORM IRD1TEST TO LOAD THE IR REGISTER
                        0011  1070 ;               i.  SET THE CONDITION CODES WITH THE TEST PATTERN ON THE WBUS
                        0011  1071 ;               j.  READ THE PSL TO THE RDM
                        0011  1072 ;       7.  TEST THE PSL Z BIT FOR CORRECT STATE
                        0011  1073 ;       8.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST PATTERNS
                        0011  1074 ;
                        0011  1075 ;
                        0011  1076 ; TEST PATTERNS:
                        0011  1077 ;
                        0011  1078 ;       ITERATION       IR      TEST PATTERN    SIZE
                        0011  1079 ;
                        0011  1080 ;           1           3E        00000000      LONG
                        0011  1081 ;           2           48        FFFFFF00      BYTE
                        0011  1082 ;           3           58        FFFFFF00      BYTE
                        0011  1083 ;           4           08        FF00FFFF      LONG
                        0011  1084 ;           5           4E        FFFF0000      WORD
                        0011  1085 ;           6           0E        FFFFFF00      BYTE
                        0011  1086 ;           7           F6        FFFFFF00      BYTE
                        0011  1087 ;           8           00        FFFFFF00      BYTE
                        0011  1088 ;           9           78        FFFFFF00      BYTE
                        0011  1089 ;           A           44        FFFFFF00      BYTE
                        0011  1090 ;           B           96        FFFFFF00      BYTE
                        0011  1091 ;           C           76        FFFFFF00      BYTE
                        0011  1092 ;           D           08        FFFFFF00      WORD
                        0011  1093 ;           E           08        FFFFFFFF      BYTE
                        0011  1094 ;           F           08        FF000000      LONG
                        0011  1095 ;          10           0A        FFFFFF00      BYTE
                        0011  1096 ;          11           0C        FFFFFF00      BYTE
                        0011  1097 ;          12           40        FFFFFF00      BYTE
                        0011  1098 ;          13           E8        FFFFFF00      BYTE
```

```
0011  1099 ;          14          08        FFFF00FF      WORD
0011  1100 ;          15          08        000000FF      LONG
0011  1101 ;          16          08        0000FF00      LONG
0011  1102 ;          17          B6        FFFFFF00      BYTE
0011  1103 ;          18          D6        FFFFFF00      BYTE
0011  1104 ;          19          EE        FFFFFF00      BYTE
0011  1105 ;          1A          F2        FFFFFF00      BYTE
0011  1106 ;          1B          F4        FFFFFF00      BYTE
0011  1107 ;
0011  1108 ;
0011  1109 ; LOGIC DESCRIPTION:
0011  1110 ;
0011  1111 ;       THESE TEST PATTERNS WERE GENERATED USING A LOGIC SIMULATOR.  SO IF
0011  1112 ;       YOU ATTEMPT TO READ THE VAX SRM AND TRY AN CORRELATE THE INSTRUCTIONS
0011  1113 ;       USED VERSUS CONDITIONS CODES IT REALLY WON'T MAKE ANY SENSE.  THE
0011  1114 ;       ATTEMPT HERE IS TO TEST THE INTERNAL WORKINGS OF THE CCC GATE ARRAY.
0011  1115 ;       ANY FAILUES SHOULD BE CONFINED TO THIS ARRAY OR THE INSTRUCTION
0011  1116 ;       REGISTER FEEDING IT.  IF YOU GET A FAILURE IN LOOP 5 AND 12 YOU MIGHT
0011  1117 ;       SUSPECT THE FPA ENABLE SIGNAL INTO THE CCC GATE ARRAY.  THIS TEST
0011  1118 ;       ATTEMPTS TO DISABLE THE FPA BEFORE IT CHECKS THE CONDITION CODES.
0011  1119 ;
0011  1120 ;
0011  1121 ; ASSUMPTIONS:
0011  1122 ;
0011  1123 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
0011  1124 ;
0011  1125 ; ERROR DESCRIPTION:
0011  1126 ;
0011  1127 ;       EXPECTED Z BIT STATE
0011  1128 ;       RECEIVED Z BIT STATE
0011  1129 ;       LOOP COUNT I
0011  1130 ;
0011  1131 ;--
0011  1132 ;*******************************************************************************
0011  1133
0011  1134
0011  1135          INITIALIZE                                  ;INT THE CPU
0013  1136          LOOP            I,1,27                      ;CREATE A TEST LOOP OF 27
001A  1137          LDFIELD         4$,I,B,6$,34,39,            ;MODIFY U-CODE FOR THIS LOOP
002B  1138          LDFIELD         5$,I,B,6$,40,41,            ;...
003C  1139          LOADDCS         6$,,<^X0C>,1                ;LOAD U-CODE INTO DCS
0043  1140          SAVEINDEX       I,3$                        ;SAVE LOOP COUNT
0048  1141          LOADREG         WDREG,3$,,L                 ;PUT LOOP COUNT INTO WDREG
0050  1142          ERRLOOP                                     ;ERROR LOOP POINT
0052  1143          FETCH           0                           ;START DCS PROGRAM
0057  1144          BRSTCLK         <^X0E>                      ;END DCS PROGRAM
005B  1145          CMPREGMSK       WHREG,1$,I,2$,,L,           ;TEST Z BIT
0067  1146          IFERROR         ,<CCC>,                     ;BIT INCORRECT
006D  1147          ENDLOOP         I                           ;END TEST LOOP
0070  1148          SKIP                                        ;GO TO NEXT TEST
0077  1149
0077  1150
0077  1151 BEGIN_TEST_DATA
0077
0077          ;-------------------------------------------------------------------------
```

```
                                  0077
                        00000004  0077  1$:     .LONG   ^X00000004                      ;RESULTS TABLE
                        00000000  007B          .LONG   ^X00000000
                        00000004  007F          .LONG   ^X00000004
                        00000000  0083          .LONG   ^X00000000
                        00000004  0087          .LONG   ^X00000004
                        00000000  008B          .LONG   ^X00000000
                        00000000  008F          .LONG   ^X00000000
                        00000000  0093          .LONG   ^X00000000
                        00000000  0097          .LONG   ^X00000000
                        00000000  009B          .LONG   ^X00000000                      ;LOOP 10
                        00000004  009F          .LONG   ^X00000004
                        00000004  00A3          .LONG   ^X00000004
                        00000000  00A7          .LONG   ^X00000000
                        00000000  00AB          .LONG   ^X00000000
                        00000000  00AF          .LONG   ^X00000000
                        00000004  00B3          .LONG   ^X00000004
                        00000004  00B7          .LONG   ^X00000004
                        00000004  00BB          .LONG   ^X00000004
                        00000004  00BF          .LONG   ^X00000004
                        00000000  00C3          .LONG   ^X00000000                      ;LOOP 20
                        00000000  00C7          .LONG   ^X00000000
                        00000000  00CB          .LONG   ^X00000000
                        00000004  00CF          .LONG   ^X00000004
                        00000004  00D3          .LONG   ^X00000004
                        00000004  00D7          .LONG   ^X00000004
                        00000004  00DB          .LONG   ^X00000004
                        00000004  00DF          .LONG   ^X00000004
                        00000004  00E3  2$:     .LONG   ^X00000004                      ;MASK FOR CMPREGMSK PSEUDO OP
                        00000000  00E7  3$:     .LONG   ^X00000000                      ;LOOP COUNT STORAGE
      01 01 01 01 01 03 02 01 01 00  00EB  4$:     .BYTE   00,01,01,02,03,01,01,01,01,01  ;RSRC LDFIELD TABLE
      06 01 01 01 01 05 04 01 01 01  00F5          .BYTE   01,01,01,04,05,01,01,01,01,06
               01 01 01 01 01 08 07  00FF          .BYTE   07,08,01,01,01,01,01
      00 00 00 00 00 01 02 00 00 02  0106  5$:     .BYTE   02,00,00,02,01,00,00,00,00,00  ;SIZE LDFIELD TABLE
      01 00 00 00 00 02 00 01 00 00  0110          .BYTE   00,00,01,00,02,00,00,00,00,01
               00 00 00 00 00 02 02  011A          .BYTE   02,02,00,00,00,00,00
                                  0121  6$:
U 0C, 7701,C800,5BE4,0201,0470,180D  0121  ;% 0C   SIZE[LONG],WB_R[TEMP0],CCOP2           ;U-CODE TO MODIFY
                                  012C
                                  012C
                                  012C  BEGIN_CPU_DATA
                                  0002
                                  0002  DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 7700,4800,5BE4,0324,2470,1802  000C  ;% 01   WCTRL/LOADCRAR,WB_R[TEMP9]             ;DISABLE ^P INTERRUPTS
U 02, 7700,4800,5BE4,0328,2C70,1803  0017  ;% 02   WCTRL/CONWRITE,WB_R[TEMP10]            ;...
U 03, 7700,C800,5BE4,03D8,0476,1804  0022  ;% 03   FPA_R[ZERO]                            ;TICK FPA OUT OF INIT STATE
U 04, 7700,4800,5BE4,03D8,2A70,1805  002D  ;% 04   WCTRL/FPA.ENABLE_WB5,WB_R[ZERO]        ;DISABLE FPA
U 05, 7700,C800,5BF4,032C,0070,1806  0038  ;% 05   PSL_R[TEMP11]                          ;IPL = 1F MODE NATIVE
U 06, 5700,8840,0364,0330,0470,1807  0043  ;% 06   R[TEMP12]_RDM                          ;LOOP COUNT TO RTEMP12
U 07, 7700,5800,EF64,0301,0470,1808  004E  ;% 07   PL_[2]                                 ;PLATCH GETS 2
U 08, 7700,0800,8B71,0330,0470,1809  0059  ;% 08   Q_R[TEMP12].RL.P                       ;FORM CACHE ADDRESS
U 09, 7700,C800,03A4,03D8,4870,180A  0064  ;% 09   WCTRL/PC_WB,WB_Q                       ;FORCE PREFETCH
U 0A, 7700,C800,0364,1700,0470,180B  006F  ;% 0A   IRD1TEST                               ;LOAD INST. REGISTER
```

```
U 0B. 7700,4800,0364,0300,0470,180C  007A  1243 ;% 0B   NOP                                    ;...
U 0C. 7701,C800,5BE4,0201,0470,180D  0085  1247 ;% 0C   SIZE[LONG],WB_R[TEMP0],CCOP2            ;SET COND. CODES
U 0D. 6700,C800,0364,0300,0870,180E  0090  1251 ;% 0D   RDM_WB,WB_PSL                          ;READ PSL TO RDM
U 0E. 7300,4800,0364,0300,0470,180F  009B  1255 ;% 0E   NOP,STOP.CLOCK                        ;STOP THE CPU CLOCK
                                     00A6  1259
                                     00A6  1260 RTEMP_DATA
                                     0001  1261
                            00000000 0001  1262         .LONG   ^X00000000                   ;TEST PATTERNS
                            FFFFFF00 0005  1263         .LONG   ^XFFFFFF00
                            FF00FFFF 0009  1264         .LONG   ^XFF00FFFF
                            FFFF0000 000D  1265         .LONG   ^XFFFF0000
                            FFFFFFFF 0011  1266         .LONG   ^XFFFFFFFF
                            FF000000 0015  1267         .LONG   ^XFF000000
                            FFFF00FF 0019  1268         .LONG   ^XFFFF00FF
                            000000FF 001D  1269         .LONG   ^X000000FF
                            0000FF00 0021  1270         .LONG   ^X0000FF00
                            00C00000 0025  1271         .LONG   ^X00C00000                   ;DISABLE ^P INTERRUPTS
                            00400000 0029  1272         .LONG   ^X00400000                   ;...
                            001F0000 002D  1273         .LONG   ^X001F0000                   ;IPL = 1F MODE NATIVE
                            00000000 0031  1274         .LONG   ^X00000000                   ;LOOP COUNT STORAGE
                                     0035  1275
                                     0035  1276 CACHE_DATA
                                     0001  1277
                            0000003E 0001  1278         .LONG   ^X0000003E                   ;INSTRUCTION TABLE
                            00000048 0005  1279         .LONG   ^X00000048
                            00000058 0009  1280         .LONG   ^X00000058
                            00000008 000D  1281         .LONG   ^X00000008
                            0000004E 0011  1282         .LONG   ^X0000004E
                            0000000E 0015  1283         .LONG   ^X0000000E
                            000000F6 0019  1284         .LONG   ^X000000F6
                            00000000 001D  1285         .LONG   ^X00000000
                            00000078 0021  1286         .LONG   ^X00000078
                            00000044 0025  1287         .LONG   ^X00000044                   ;LOOP 10
                            00000096 0029  1288         .LONG   ^X00000096
                            00000076 002D  1289         .LONG   ^X00000076
                            00000008 0031  1290         .LONG   ^X00000008
                            00000008 0035  1291         .LONG   ^X00000008
                            00000008 0039  1292         .LONG   ^X00000008
                            0000000A 003D  1293         .LONG   ^X0000000A
                            0000000C 0041  1294         .LONG   ^X0000000C
                            00000040 0045  1295         .LONG   ^X00000040
                            000000E8 0049  1296         .LONG   ^X000000E8
                            00000008 004D  1297         .LONG   ^X00000008                   ;LOOP 20
                            00000008 0051  1298         .LONG   ^X00000008
                            00000008 0055  1299         .LONG   ^X00000008
                            000000B6 0059  1300         .LONG   ^X000000B6
                            000000D6 005D  1301         .LONG   ^X000000D6
                            000000EE 0061  1302         .LONG   ^X000000EE
                            000000F2 0065  1303         .LONG   ^X000000F2
                            000000F4 0069  1304         .LONG   ^X000000F4
                            00000000 006D  1305         .LONG   ^X00000000
                                     0071  1306
                                     0071  1307 END_CPU_DATA
                                     012C  1308
                                     012C  1309
```

```
                012C  1310 END_TEST_DATA
                012C
                012C       ;-----------------------------------------------------------------------------
                012C  1311 ENDTEST
```

```
                        0011        .SBTTL  "TEST   094     Z BIT TEST 2
                        0011  1313 ;*******************************************************************************
                        0011  1314 ;++
                        0011  1315 ;
                        0011  1316 ; FUNCTIONAL DESCRIPTION:
                        0011  1317 ;
                        0011  1318 ;       THIS TEST VERIFIES THAT THE Z BIT IN THE CCC GATE ARRAY SETS PROPERLY
                        0011  1319 ;       FOR COMPATABILITY MODE INSTRUCTIONS USING CC/CCOP2.CCBR_SIGND
                        0011  1320 ;       MICRO ORDER.
                        0011  1321 ;
                        0011  1322 ;
                        0011  1323 ; TEST ALGORITHM:
                        0011  1324 ;
                        0011  1325 ;       1.  CREATE A TEST LOOP OF 12
                        0011  1326 ;       2.  MODIFY THE MICRO CODE FOR EACH LOOP THROUGH THE TEST
                        0011  1327 ;       3.  LOAD THE NEW MICRO CODE INTO THE RDM DCS
                        0011  1328 ;       4.  SAVE THE LOOP COUNT
                        0011  1329 ;       5.  MOVE THE LOOP COUNT TO THE RDM WDREG
                        0011  1330 ;       6.  EXECUTE THE DCS PROGRAM
                        0011  1331 ;               a.  DISABLE ^P INTERRUPTS
                        0011  1332 ;               b.  SET THE IPL TO 1F AND MODE TO NATIVE
                        0011  1333 ;               c.  MOVE THE LOOP COUNT TO THE RTEMPS
                        0011  1334 ;               d.  LOAD THE PLATCH WITH 2
                        0011  1335 ;               e.  ROTATE THE LOOP COUNT LEFT BY 2 TO FORM A CACHE ADDRESS
                        0011  1336 ;               f.  LOAD THE PC WITH THE CACHE ADDRESS TO FORCE PREFETCH
                        0011  1337 ;               g.  PERFORM IRD1TEST TO LOAD THE IR REGISTER
                        0011  1338 ;               h.  SET IPL = 1F AND COMPATABILITY MODE
                        0011  1339 ;               i.  SET CONDITION CODES WITH TEST PATTERN ON WBUS
                        0011  1340 ;               j.  READ THE PSL TO THE RDM
                        0011  1341 ;       7.  TEST THE PSL N BIT FOR CORRECT STATE
                        0011  1342 ;       8.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST PATTERNS
                        0011  1343 ;
                        0011  1344 ;
                        0011  1345 ; TEST PATTERNS:
                        0011  1346 ;
                        0011  1347 ;       ITERATION       IR      TEST PATTERN    SIZE
                        0011  1348 ;
                        0011  1349 ;           1           22        00000000      LONG
                        0011  1350 ;           2           00        FFFFFF00      WORD
                        0011  1351 ;           3           04        FFFFFF00      BYTE
                        0011  1352 ;           4           00        00FF0000      LONG
                        0011  1353 ;           5           20        FFFFFF00      BYTE
                        0011  1354 ;           6           00        FFFF0000      WORD
                        0011  1355 ;           7           26        FFFFFF00      BYTE
                        0011  1356 ;           8           00        FFFFFFFF      BYTE
                        0011  1357 ;           9           00        FF000000      LONG
                        0011  1358 ;           A           00        FFFF00FF      WORD
                        0011  1359 ;           B           00        000000FF      LONG
                        0011  1360 ;           C           00        0000FF00      LONG
                        0011  1361 ;
                        0011  1362 ;
                        0011  1363 ; TEST ALGORITHM:
                        0011  1364 ;
                        0011  1365 ;       THESE TEST PATTERNS WERE GENERATED USING A LOGIC SIMULATOR.  SO IF
                        0011  1366 ;       YOU ATTEMPT TO READ THE VAX SRM AND TRY AN CORRELATE THE INSTRUCTIONS
```

```
                                0011  ;       USED VERSUS CONDITIONS CODES IT REALLY WON'T MAKE ANY SENSE.  THE
                                0011  ;       ATTEMPT HERE IS TO TEST THE INTERNAL WORKINGS OF THE CCC GATE ARRAY.
                                0011  ;       ANY FAILUES SHOULD BE CONFINED TO THIS ARRAY OR THE INSTRUCTION
                                0011  ;       REGISTER FEEDING IT.
                                0011  ;
                                0011  ; ASSUMPTIONS:
                                0011  ;
                                0011  ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
                                0011  ;
                                0011  ; ERROR DESCRIPTION:
                                0011  ;
                                0011  ;       EXPECTED Z BIT STATE
                                0011  ;       RECEIVED Z BIT STATE
                                0011  ;       LOOP COUNT I
                                0011  ;
                                0011  ;--
                                0011  ;*******************************************************************************
                                0011
                                0011
                                0011          INITIALIZE                                  ;INT THE CPU
                                0013          LOOP            I,1,12                      ;CREATE A TEST LOOP OF 12
                                001A          LDFIELD         4$,I,B,6$,34,39,            ;MODIFY U-CODE FOR THIS LOOP
                                002B          LDFIELD         5$,I,B,6$,40,41,            ;...
                                003C          LOADDCS         6$,,<^X0A>,1                ;LOAD U-CODE INTO DCS
                                0043          SAVEINDEX       I,3$                        ;SAVE LOOP COUNT
                                0048          LOADREG         WDREG,3$,,L                 ;PUT LOOP COUNT INTO WDREG
                                0050          ERRLOOP                                     ;ERROR LOOP POINT
                                0052          FETCH           0                           ;START DCS PROGRAM
                                0057          BRSTCLK         <^X0C>                      ;END DCS PROGRAM
                                005B          CMPREGMSK       WHREG,1$,I,2$,,L,           ;TEST Z BIT
                                0067          IFERROR         ,<CCC>,                     ;BIT INCORRECT
                                006D          ENDLOOP         I                           ;END TEST LOOP
                                0070          SKIP                                        ;GO TO NEXT TEST
                                0077
                                0077
                                0077  BEGIN_TEST_DATA
                                0077
                                0077  ;-----------------------------------------------------------------------
                                0077
                      00000004  0077  1$:     .LONG   ^X00000004                          ;RESULTS TABLE
                      00000000  007B          .LONG   ^X00000000
                      00000004  007F          .LONG   ^X00000004
                      00000000  0083          .LONG   ^X00000000
                      00000000  0087          .LONG   ^X00000000
                      00000004  008B          .LONG   ^X00000004
                      00000000  008F          .LONG   ^X00000000
                      00000000  0093          .LONG   ^X00000000
                      00000000  0097          .LONG   ^X00000000
                      00000000  009B          .LONG   ^X00000000                          ;LOOP 10
                      00000000  009F          .LONG   ^X00000000
                      00000000  00A3          .LONG   ^X00000000
                      00000004  00A7  2$:     .LONG   ^X00000004                          ;MASK FOR CMPREGMSK PSEUDO OP
                      00000000  00AB  3$:     .LONG   ^X00000000                          ;LOOP COUNT STORAGE
06 05 04 01 03 01 02 01 01 00   00AF  4$:     .BYTE   00,01,01,02,01,03,01,04,05,06       ;RSRC LDFIELD TABLE
                         08 07  00B9          .BYTE   07,08
```

```
                                00BB  1420
      01 02 00 00 01 00 02 00 01 02  00BB  1421 5$:     .BYTE   02,01,00,02,00,01,00,00,02,01   ;SIZE LDFIELD TABLE
                                  02 02  00C5  1422         .BYTE   02,02
                                00C7  1423 6$:
U 0A. 7701,C800,5BE4,0201,0470,180B  00C7  1424 ;% 0A   SIZE[LONG],WB_R[TEMP0],CCOP2          ;U-CODE TO MODIFY
                                00D2  1428
                                00D2  1429
                                00D2  1430 BEGIN_CPU_DATA
                                0002  1431
                                0002  1432 DCS_DATA
                                0001  1433
U 00. 7700,C800,0364,0300,0470,1801  0001  1434 ;% 00   NOP
U 01. 7700,4800,5BE4,0324,2470,1802  000C  1438 ;% 01   WCTRL/LOADCRAR,WB_R[TEMP9]             ;DISABLE ^P INTERRUPTS
U 02. 7700,4800,5BE4,0328,2C70,1803  0017  1442 ;% 02   WCTRL/CONWRITE,WB_R[TEMP10]           ;...
U 03. 7700,4800,5BF4,032C,0070,1804  0022  1446 ;% 03   PSL_R[TEMP11]                        ;IPL = 1F NATIVE MODE
U 04. 5700,4840,0364,0334,0470,1805  002D  1450 ;% 04   R[TEMP13]_RDM                        ;LOOP COUNT TO RTEMP13
U 05. 7700,D800,EF64,0301,0470,1806  0038  1454 ;% 05   PL_[2]                              ;PLATCH GETS 2
U 06. 7700,C800,8B71,0334,0470,1807  0043  1458 ;% 06   Q_R[TEMP13].RL.P                    ;FORM CACHE ADDRESS
U 07. 7700,4800,03A4,03D8,4870,1808  004E  1462 ;% 07   WCTRL/PC_WB,WB_Q                     ;FORCE PREFETCH
U 08. 7700,4800,0364,1700,0470,1809  0059  1466 ;% 08   IRD1TEST                            ;LOAD INST. REGISTER
U 09. 7700,8800,5BF4,0330,0070,180A  0064  1470 ;% 09   PSL_R[TEMP12]                       ;IPL = 1F COMPATABILITY MODE
U 0A. 7701,C800,5BE4,0201,0470,180B  006F  1474 ;% 0A   SIZE[LONG],WB_R[TEMP0],CCOP2          ;SET COND. CODES
U 0B. 6700,4800,0364,0300,0870,180C  007A  1478 ;% 0B   RDM_WB,WB_PSL                        ;READ PSL TO RDM
U 0C. 7300,C800,0364,0300,0470,180D  0085  1482 ;% 0C   NOP,STOP.CLOCK                      ;STOP THE CPU CLOCK
                                0090  1486
                                0090  1487 RTEMP_DATA
                                0001  1488
                    00000000    0001  1489         .LONG   ^X00000000                      ;TEST PATTERNS
                    FFFFFF00    0005  1490         .LONG   ^XFFFFFF00
                    00FF0000    0009  1491         .LONG   ^X00FF0000
                    FFFF0000    000D  1492         .LONG   ^XFFFF0000
                    FFFFFFFF    0011  1493         .LONG   ^XFFFFFFFF
                    FF000000    0015  1494         .LONG   ^XFF000000
                    FFFF00FF    0019  1495         .LONG   ^XFFFF00FF
                    000000FF    001D  1496         .LONG   ^X000000FF
                    0000FF00    0021  1497         .LONG   ^X0000FF00
                    00C00000    0025  1498         .LONG   ^X00C00000                      ;DISABLE ^P INTERRUPTS
                    00400000    0029  1499         .LONG   ^X00400000                      ;...
                    001F0000    002D  1500         .LONG   ^X001F0000                      ;IPL = 1F NATIVE MODE
                    801F0000    0031  1501         .LONG   ^X801F0000                      ;IPL = IF COMPATABILITY MODE
                    00000000    0035  1502         .LONG   ^X00000000                      ;LOOP COUNT STORAGE
                                0039  1503
                                0039  1504 CACHE_DATA
                                0001  1505
                    00000022    0001  1506         .LONG   ^X00000022                      ;INSTRUCTION TABLE
                    00000000    0005  1507         .LONG   ^X00000000
                    00000004    0009  1508         .LONG   ^X00000004
                    00000000    000D  1509         .LONG   ^X00000000
                    00000020    0011  1510         .LONG   ^X00000020
                    00000000    0015  1511         .LONG   ^X00000000
                    00000026    0019  1512         .LONG   ^X00000026
                    00000000    001D  1513         .LONG   ^X00000000
                    00000000    0021  1514         .LONG   ^X00000000
                    00000000    0025  1515         .LONG   ^X00000000                      ;LOOP 10
                    00000000    0029  1516         .LONG   ^X00000000
```

```
00000000  002D  1517          .LONG   ^X00000000
00000000  0031  1518          .LONG   ^X00000000
          0035  1519
          0035  1520 END_CPU_DATA
          00D2  1521
          00D2  1522
          00D2  1523 END_TEST_DATA
          00D2
          00D2          ;-----------------------------------------------------------------------------
          00D2  1524 ENDTEST
```

```
                0011        .SBTTL  "TEST   095     V BIT TEST 1
                0011   1526 ;*******************************************************************************
                0011   1527 ;++
                0011   1528 ;
                0011   1529 ; FUNCTIONAL DESCRIPTION:
                0011   1530 ;
                0011   1531 ;       THIS TEST VERIFIES THAT THE V BIT IN THE CCC GATE ARRAY SETS PROPERLY
                0011   1532 ;       FOR NATIVE MODE INSTRUCTIONS USING CC/CCOP2.CCBR_SIGND MICRO ORDER.
                0011   1533 ;
                0011   1534 ;
                0011   1535 ; TEST ALGORITHM:
                0011   1536 ;
                0011   1537 ;       1.  CREATE A TEST LOOP OF 25
                0011   1538 ;       2.  MODIFY THE MICRO CODE FOR EACH LOOP THROUGH THE TEST
                0011   1539 ;       3.  LOAD THE NEW MICRO CODE INTO THE RDM DCS
                0011   1540 ;       4.  SAVE THE LOOP COUNT
                0011   1541 ;       5.  MOVE THE LOOP COUNT TO THE RDM WDREG
                0011   1542 ;       6.  EXECUTE THE DCS PROGRAM
                0011   1543 ;               a.  DISABLE ^P INTERRUPTS
                0011   1544 ;               b.  SET THE IPL TO 1F AND MODE TO NATIVE
                0011   1545 ;               c.  MOVE THE LOOP COUNT TO THE RTEMPS
                0011   1546 ;               d.  LOAD THE PLATCH WITH 2
                0011   1547 ;               e.  ROTATE THE LOOP COUNT LEFT BY 2 TO FORM A CACHE ADDRESS
                0011   1548 ;               f.  LOAD THE PC WITH THE CACHE ADDRESS TO FORCE PREFETCH
                0011   1549 ;               g.  PERFORM IRD1TEST TO LOAD THE IR REGISTER
                0011   1550 ;               h.  LOAD THE Q REGISTER WITH TEST DATA
                0011   1551 ;               i.  ADD THE Q REGISTER TO ONE OF THE RTEMPS TO CREATE THE
                0011   1552 ;                       TEST CONDITIONS WHILE PERFORMING CCOP2.
                0011   1553 ;               j.  READ THE PSL TO THE RDM
                0011   1554 ;       7.  TEST THE PSL V BIT FOR CORRECT STATE
                0011   1555 ;       8.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST PATTERNS
                0011   1556 ;
                0011   1557 ;
                0011   1558 ; TEST PATTERNS:
                0011   1559 ;
                0011   1560 ;       ITERATION       IR      Q REG. + R[TEMP*]       SIZE
                0011   1561 ;
                0011   1562 ;           1           1E      80FF0080 + 80000080     WORD
                0011   1563 ;           2           06      00000000 + 00000000     BYTE
                0011   1564 ;           3           00      00000080 + 00000080     BYTE
                0011   1565 ;           4           3E      00000080 + 00000080     BYTE
                0011   1566 ;           5           F6      000000FF + 00000000     BYTE
                0011   1567 ;           6           18      40404040 + 40000000     LONG
                0011   1568 ;           7           58      40404040 + 00004040     LONG
                0011   1569 ;           8           48      000000FF + 00000000     BYTE
                0011   1570 ;           9           F0      40404040 + 00004000     WORD
                0011   1571 ;           A           86      00000000 + 00000000     BYTE
                0011   1572 ;           B           00      FF000000 + 00000000     BYTE
                0011   1573 ;           C           44      0000FF00 + 00000000     BYTE
                0011   1574 ;           D           60      FFFFFFFF + 00000000     BYTE
                0011   1575 ;           E           08      40404040 + 00000040     BYTE
                0011   1576 ;           F           12      00000000 + 00000000     BYTE
                0011   1577 ;          10           B4      40404040 + 00000040     BYTE
                0011   1578 ;          11           EE      40404040 + 00000040     BYTE
                0011   1579 ;          12           1C      00004040 + 00004040     LONG
```

```
                     0011  1580 ;          13          D8      40404040 + 00000040     BYTE
                     0011  1581 ;          14          4C      FFFFFFFF + 00000000     BYTE
                     0011  1582 ;          15          74      40404040 + 00000040     BYTE
                     0011  1583 ;          16          46      00000000 + 00000000     BYTE
                     0011  1584 ;          17          C8      FFFFFFFF + 00000000     BYTE
                     0011  1585 ;          18          F2      FFFFFFFF + 00000000     BYTE
                     0011  1586 ;          19          F4      FFFFFFFF + 00000000     BYTE
                     0011  1587 ;
                     0011  1588 ;
                     0011  1589 ; TEST ALGORITHM:
                     0011  1590 ;
                     0011  1591 ;       THESE TEST PATTERNS WERE GENERATED USING A LOGIC SIMULATOR.  SO IF
                     0011  1592 ;       YOU ATTEMPT TO READ THE VAX SRM AND TRY AN CORRELATE THE INSTRUCTIONS
                     0011  1593 ;       USED VERSUS CONDITIONS CODES IT REALLY WON'T MAKE ANY SENSE.  THE
                     0011  1594 ;       ATTEMPT HERE IS TO TEST THE INTERNAL WORKINGS OF THE CCC GATE ARRAY.
                     0011  1595 ;       ANY FAILUES SHOULD BE CONFINED TO THIS ARRAY OR THE INSTRUCTION
                     0011  1596 ;       REGISTER FEEDING IT.
                     0011  1597 ;
                     0011  1598 ; ASSUMPTIONS:
                     0011  1599 ;
                     0011  1600 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
                     0011  1601 ;
                     0011  1602 ; ERROR DESCRIPTION:
                     0011  1603 ;
                     0011  1604 ;       EXPECTED V BIT STATE
                     0011  1605 ;       RECEIVED V BIT STATE
                     0011  1606 ;       LOOP COUNT I
                     0011  1607 ;
                     0011  1608 ;--
                     0011  1609 ;*******************************************************************************
                     0011  1610
                     0011  1611
                     0011  1612          INITIALIZE                              ;INT THE CPU
                     0013  1613          LOOP            I,1,25                  ;CREATE A TEST LOOP OF 25
                     001A  1614          LDFIELD         4$,I,B,7$,34,39,        ;MODIFY U-CODE FOR THIS LOOP
                     002B  1615          LDFIELD         5$,I,B,8$,40,41,        ;...
                     003C  1616          LDFIELD         6$,I,B,8$,34,39,        ;...
                     004D  1617          LOADDCS         7$,,<^X09>,2            ;LOAD U-CODE INTO DCS
                     0054  1618          SAVEINDEX       I,3$                    ;SAVE LOOP COUNT
                     0059  1619          LOADREG         WDREG,3$,,L             ;PUT LOOP COUNT INTO WDREG
                     0061  1620          ERRLOOP                                 ;ERROR LOOP POINT
                     0063  1621          FETCH           0                       ;START DCS PROGRAM
                     0068  1622          BRSTCLK         <^X0C>                  ;END DCS PROGRAM
                     006C  1623          CMPREGMSK       WHREG,1$,I,2$,,L,       ;TEST V BIT
                     0078  1624          IFERROR         ,<CCC>,                 ;BIT INCORRECT
                     007E  1625          ENDLOOP         I                       ;END TEST LOOP
                     0081  1626          SKIP                                    ;GO TO NEXT TEST
                     0088  1627
                     0088  1628
                     0088  1629 BEGIN_TEST_DATA
                     0088
                     0088          ;-----------------------------------------------------------------------
                     0088  1630
00000000             0088  1631 1$:       .LONG   ^X00000000                      ;RESULTS TABLE
00000002             008C  1632          .LONG   ^X00000002
```

```
                        00000002  0090  1633          .LONG   ^X00000002
                        00000000  0094  1634          .LONG   ^X00000000
                        00000002  0098  1635          .LONG   ^X00000002
                        00000002  009C  1636          .LONG   ^X00000002
                        00000000  00A0  1637          .LONG   ^X00000000
                        00000002  00A4  1638          .LONG   ^X00000002
                        00000002  00A8  1639          .LONG   ^X00000002
                        00000000  00AC  1640          .LONG   ^X00000000                              ;LOOP 10
                        00000002  00B0  1641          .LONG   ^X00000002
                        00000002  00B4  1642          .LONG   ^X00000002
                        00000000  00B8  1643          .LONG   ^X00000000
                        00000000  00BC  1644          .LONG   ^X00000000
                        00000000  00C0  1645          .LONG   ^X00000000
                        00000000  00C4  1646          .LONG   ^X00000000
                        00000000  00C8  1647          .LONG   ^X00000000
                        00000000  00CC  1648          .LONG   ^X00000000
                        00000000  00D0  1649          .LONG   ^X00000000
                        00000000  00D4  1650          .LONG   ^X00000000                              ;LOOP 20
                        00000000  00D8  1651          .LONG   ^X00000000
                        00000000  00DC  1652          .LONG   ^X00000000
                        00000000  00E0  1653          .LONG   ^X00000000
                        00000000  00E4  1654          .LONG   ^X00000000
                        00000000  00E8  1655          .LONG   ^X00000000
                        00000002  00EC  1656 2$:      .LONG   ^X00000002                              ;MASK FOR CMPREGMSK PSEUDO OP
                        00000000  00F0  1657 3$:      .LONG   ^X00000000                              ;LOOP COUNT STORAGE
    01 04 03 04 04 03 02 02 01 00   00F4  1658 4$:      .BYTE   00,01,02,02,03,04,04,03,04,01           ;RSRC LDFIELD TABLE
    07 04 08 04 04 01 04 07 06 05   00FE  1659          .BYTE   05,06,07,04,01,04,04,08,04,07
                07 07 07 01 04      0108  1660          .BYTE   04,01,07,07,07
                                    010D  1661
    00 01 00 02 02 00 00 00 00 01   010D  1662 5$:      .BYTE   01,00,00,00,00,02,02,00,01,00           ;SIZE LDFIELD TABLE
    00 00 02 00 00 00 00 00 00 00   0117  1663          .BYTE   00,00,00,00,00,00,00,02,00,00
                00 00 00 00 00      0121  1664          .BYTE   00,00,00,00,00
                                    0126  1665
    01 0B 01 08 0A 01 02 02 01 09   0126  1666 6$:      .BYTE   09,01,02,02,01,10,08,01,11,01           ;RSRC LDFIELD TABLE
    01 0C 08 0C 0C 01 0C 01 01 01   0130  1667          .BYTE   01,01,01,12,01,12,12,08,12,01
                01 01 01 01 0C      013A  1668          .BYTE   12,01,01,01,01
                                    013F  1669 7$:
U 09, 7700,0800,5BE5,0300,0470,180A 013F  1670 ;% 09    Q_R[TEMP0]                                      ;U-CODE TO MODIFY FOR EACH
                                    014A  1674 8$:                                                      ;LOOP
U 0A, 7701,8800,0390,0125,0470,180B 014A  1675 ;% 0A    SIZE[WORD],ALU/A+B+CI,MUX/R,Q,RSRC/TEMP9,CCOP2
                                    0155  1679
                                    0155  1680
                                    0155  1681 BEGIN_CPU_DATA
                                    0002  1682
                                    0002  1683 DCS_DATA
                                    0001  1684
U 00, 7700,C800,0364,0300,0470,1801 0001  1685 ;% 00    NOP
U 01, 7700,0800,5BE4,0334,2470,1802 000C  1689 ;% 01    WCTRL/LOADCRAR,WB_R[TEMP13]                     ;DISABLE ^P INTERRUPTS
U 02, 7700,0800,5BE4,0338,2C70,1803 0017  1693 ;% 02    WCTRL/CONWRITE,WB_R[MM.TEMP5]                   ;...
U 03, 7700,0800,5BF4,033C,0070,1804 0022  1697 ;% 03    PSL_R[MM.TEMP1]                                 ;SET IPL = 1F AND NATIVE MODE
U 04, 5700,4840,0364,0340,0470,1805 002D  1701 ;% 04    R[R0]_RDM                                       ;LOOP COUNT TO RTEMPS
U 05, 7700,D800,EF64,0301,0470,1806 0038  1705 ;% 05    PL_[2]                                          ;PLATCH GETS 2
U 06, 7700,C800,8B71,0340,0470,1807 0043  1709 ;% 06    Q_R[R0].RL.P                                    ;FORM CACHE ADDRESS
U 07, 7700,4800,03A4,03D8,4870,1808 004E  1713 ;% 07    WCTRL/PC_WB,WB_Q                                ;FORCE PREFETCH
U 08, 7700,4800,0364,1700,0470,1809 0059  1717 ;% 08    IRD1TEST                                        ;LOAD INST. REGISTER
```

```
U 09, 7700,0800,5BE5,0300,0470,180A  0064  1721 ;% 09   Q_R[TEMP0]                                    ;TEST PATTERN TO Q REGISTER
U 0A, 7701,8800,0390,0125,0470,180B  006F  1725 ;% 0A   SIZE[WORD],ALU/A+B+CI,MUX/R.Q,RSRC/TEMP9,CCOP2;SET COND. CODES
U 0B, 6700,4800,0364,0300,0870,180C  007A  1729 ;% 0B   RDM_WB,WB_PSL                                ;READ PSL TO RDM
U 0C, 7300,C800,0364,0300,0470,180D  0085  1733 ;% 0C   NOP,STOP.CLOCK                               ;STOP THE CPU CLOCK
                                     0090  1737
                                     0090  1738 RTEMP_DATA
                                     0001  1739
                           80FF0080  0001  1740          .LONG   ^X80FF0080                         ;TEST PATTERNS
                           00000000  0005  1741          .LONG   ^X00000000
                           00000080  0009  1742          .LONG   ^X00000080
                           000000FF  000D  1743          .LONG   ^X000000FF
                           40404040  0011  1744          .LONG   ^X40404040
                           FF000000  0015  1745          .LONG   ^XFF000000
                           0000FF00  0019  1746          .LONG   ^X0000FF00
                           FFFFFFFF  001D  1747          .LONG   ^XFFFFFFFF
                           00004040  0021  1748          .LONG   ^X00004040
                           80000080  0025  1749          .LONG   ^X80000080
                           40000000  0029  1750          .LONG   ^X40000000
                           00004000  002D  1751          .LONG   ^X00004000
                           00000040  0031  1752          .LONG   ^X00000040
                           00C00000  0035  1753          .LONG   ^X00C00000                         ;DISABLE ^P INTERRUPTS
                           00400000  0039  1754          .LONG   ^X00400000                         ;...
                           001F0000  003D  1755          .LONG   ^X001F0000                         ;IPL = 1F NATIVE MODE
                           00000000  0041  1756          .LONG   ^X00000000                         ;LOOP COUNT STORAGE
                                     0045  1757
                                     0045  1758 CACHE_DATA
                                     0001  1759
                  0000001E           0001  1760          .LONG   ^X0000001E                         ;INSTRUCTION TABLE
                  00000006           0005  1761          .LONG   ^X00000006
                  00000000           0009  1762          .LONG   ^X00000000
                  0000003E           000D  1763          .LONG   ^X0000003E
                  000000F6           0011  1764          .LONG   ^X000000F6
                  00000018           0015  1765          .LONG   ^X00000018
                  00000058           0019  1766          .LONG   ^X00000058
                  00000048           001D  1767          .LONG   ^X00000048
                  000000F0           0021  1768          .LONG   ^X000000F0
                  00000086           0025  1769          .LONG   ^X00000086                         ;LOOP 10
                  00000000           0029  1770          .LONG   ^X00000000
                  00000044           002D  1771          .LONG   ^X00000044
                  00000060           0031  1772          .LONG   ^X00000060
                  00000008           0035  1773          .LONG   ^X00000008
                  00000012           0039  1774          .LONG   ^X00000012
                  000000B4           003D  1775          .LONG   ^X000000B4
                  000000EE           0041  1776          .LONG   ^X000000EE
                  0000001C           0045  1777          .LONG   ^X0000001C
                  000000D8           0049  1778          .LONG   ^X000000D8
                  0000004C           004D  1779          .LONG   ^X0000004C                         ;LOOP 20
                  00000074           0051  1780          .LONG   ^X00000074
                  00000046           0055  1781          .LONG   ^X00000046
                  000000C8           0059  1782          .LONG   ^X000000C8
                  000000F2           005D  1783          .LONG   ^X000000F2
                  000000F4           0061  1784          .LONG   ^X000000F4
                  00000000           0065  1785          .LONG   ^X00000000
                                     0069  1786
                                     0069  1787 END_CPU_DATA
```

```
                     0155  1788
                     0155  1789
                     0155  1790 END_TEST_DATA
                     0155
                     0155       ;-----------------------------------------------------------------------------
                     0155  1791 ENDTEST
```

```
                        0011        .SBTTL  "TEST   096     V BIT TEST 2
                        0011  1793 ;*******************************************************************************
                        0011  1794 ;++
                        0011  1795 ;
                        0011  1796 ; FUNCTIONAL DESCRIPTION:
                        0011  1797 ;
                        0011  1798 ;       THIS TEST VERIFIES THAT THE V BIT IN THE CCC GATE ARRAY SETS PROPERLY
                        0011  1799 ;       FOR COMPATABILITY MODE INSTRUCTIONS USING CC/CCOP2.CCBR_SIGND
                        0011  1800 ;       MICRO ORDER.
                        0011  1801 ;
                        0011  1802 ;
                        0011  1803 ; TEST ALGORITHM:
                        0011  1804 ;
                        0011  1805 ;       1.  CREATE A TEST LOOP OF 17
                        0011  1806 ;       2.  MODIFY THE MICRO CODE FOR EACH LOOP THROUGH THE TEST
                        0011  1807 ;       3.  LOAD THE NEW MICRO CODE INTO THE RDM DCS
                        0011  1808 ;       4.  SAVE THE LOOP COUNT
                        0011  1809 ;       5.  MOVE THE LOOP COUNT TO THE RDM WDREG
                        0011  1810 ;       6.  EXECUTE THE DCS PROGRAM
                        0011  1811 ;               a.  DISABLE ^P INTERRUPTS
                        0011  1812 ;               b.  SET THE IPL TO 1F AND MODE TO NATIVE
                        0011  1813 ;               c.  MOVE THE LOOP COUNT TO THE RTEMPS
                        0011  1814 ;               d.  LOAD THE PLATCH WITH 2
                        0011  1815 ;               e.  ROTATE THE LOOP COUNT LEFT BY 2 TO FORM A CACHE ADDRESS
                        0011  1816 ;               f.  LOAD THE PC WITH THE CACHE ADDRESS TO FORCE PREFETCH
                        0011  1817 ;               g.  PERFORM IRD1TEST TO LOAD THE IR REGISTER
                        0011  1818 ;               h.  LOAD THE Q REGISTER WITH TEST DATA
                        0011  1819 ;               i.  SET IPL TO 1F AND COMPATABILITY MODE
                        0011  1820 ;               h.  ADD THE Q REGISTER TO ONE OF THE RTEMPS TO CREATE THE
                        0011  1821 ;                       TEST CONDITIONS WHILE PERFORMING CCOP2.
                        0011  1822 ;               j.  READ THE PSL TO THE RDM
                        0011  1823 ;       7.  TEST THE PSL V BIT FOR CORRECT STATE
                        0011  1824 ;       8.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST PATTERNS
                        0011  1825 ;
                        0011  1826 ;
                        0011  1827 ; TEST PATTERNS:
                        0011  1828 ;
                        0011  1829 ;       ITERATION       IR      Q REG. + R[TEMP*]       SIZE
                        0011  1830 ;
                        0011  1831 ;           1           24      00000000 + 00000000     BYTE
                        0011  1832 ;           2           6C      00404040 + 00004040     LONG
                        0011  1833 ;           3           0C      0000C040 + 0000C040     WORD
                        0011  1834 ;           4           84      40000040 + 00000040     BYTE
                        0011  1835 ;           5           E4      80000000 + 80000000     LONG
                        0011  1836 ;           6           00      00008000 + 00008000     WORD
                        0011  1837 ;           7           64      80400000 + 80000000     LONG
                        0011  1838 ;           8           24      80004000 + 80000000     LONG
                        0011  1839 ;           9           EA      00000080 + 00000080     BYTE
                        0011  1840 ;           A           84      40408040 + 00000000     WORD
                        0011  1841 ;           B           26      00000040 + 00000040     BYTE
                        0011  1842 ;           C           64      00000040 + 00000000     WORD
                        0011  1843 ;           D           4C      40404040 + 00000000     LONG
                        0011  1844 ;           E           84      80000000 + 00000000     LONG
                        0011  1845 ;           F           84      80000080 + 00004080     BYTE
                        0011  1846 ;          10           28      40404040 + 00000040     BYTE
```

```
                     0011  1847 ;          11           24      00008080 + 00008080     LONG
                     0011  1848 ;
                     0011  1849 ;
                     0011  1850 ; TEST ALGORITHM:
                     0011  1851 ;
                     0011  1852 ;       THESE TEST PATTERNS WERE GENERATED USING A LOGIC SIMULATOR.  SO IF
                     0011  1853 ;       YOU ATTEMPT TO READ THE VAX SRM AND TRY AN CORRELATE THE INSTRUCTIONS
                     0011  1854 ;       USED VERSUS CONDITIONS CODES IT REALLY WON'T MAKE ANY SENSE.  THE
                     0011  1855 ;       ATTEMPT HERE IS TO TEST THE INTERNAL WORKINGS OF THE CCC GATE ARRAY.
                     0011  1856 ;       ANY FAILUES SHOULD BE CONFINED TO THIS ARRAY OR THE INSTRUCTION
                     0011  1857 ;       REGISTER FEEDING IT.
                     0011  1858 ;
                     0011  1859 ; ASSUMPTIONS:
                     0011  1860 ;
                     0011  1861 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
                     0011  1862 ;
                     0011  1863 ; ERROR DESCRIPTION:
                     0011  1864 ;
                     0011  1865 ;       EXPECTED V BIT STATE
                     0011  1866 ;       RECEIVED V BIT STATE
                     0011  1867 ;       LOOP COUNT I
                     0011  1868 ;
                     0011  1869 ;--
                     0011  1870 ;*******************************************************************************
                     0011  1871
                     0011  1872
                     0011  1873          INITIALIZE                                  ;INT THE CPU
                     0013  1874          LOOP            I,1,17                      ;CREATE A TEST LOOP OF 17
                     001A  1875          LDFIELD         4$,I,B,7$,34,39,            ;MODIFY U-CODE FOR THIS LOOP
                     002B  1876          LDFIELD         5$,I,B,8$,40,41,            ;...
                     003C  1877          LDFIELD         6$,I,B,8$,34,39,            ;...
                     004D  1878          LOADDCS         7$,,<^X09>,3                ;LOAD U-CODE INTO DCS
                     0054  1879          SAVEINDEX       I,3$                        ;SAVE LOOP COUNT
                     0059  1880          LOADREG         WDREG,3$,,L                 ;PUT LOOP COUNT INTO WDREG
                     0061  1881          ERRLOOP                                     ;ERROR LOOP POINT
                     0063  1882          FETCH           0                           ;START DCS PROGRAM
                     0068  1883          BRSTCLK         <^X0D>                      ;END DCS PROGRAM
                     006C  1884          CMPREGMSK       WHREG,1$,I,2$,,L,           ;TEST V BIT
                     0078  1885          IFERROR         ,<CCC>,                     ;BIT INCORRECT
                     007E  1886          ENDLOOP         I                           ;END TEST LOOP
                     0081  1887          SKIP                                        ;GO TO NEXT TEST
                     0088  1888
                     0088  1889
                     0088  1890 BEGIN_TEST_DATA
                     0088
                     0088              ;-------------------------------------------------------------------------
                     0088  1891
00000000             0088  1892 1$:       .LONG   ^X00000000                        ;RESULTS TABLE
00000002             008C  1893          .LONG   ^X00000002
00000000             0090  1894          .LONG   ^X00000000
00000000             0094  1895          .LONG   ^X00000000
00000002             0098  1896          .LONG   ^X00000002
00000002             009C  1897          .LONG   ^X00000002
00000000             00A0  1898          .LONG   ^X00000000
00000000             00A4  1899          .LONG   ^X00000000
```

```
                          00000002  00A8  1900          .LONG   ^X00000002
                          00000000  00AC  1901          .LONG   ^X00000000                      ;LOOP 10
                          00000000  00B0  1902          .LONG   ^X00000000
                          00000002  00B4  1903          .LONG   ^X00000002
                          00000002  00B8  1904          .LONG   ^X00000002
                          00000002  00BC  1905          .LONG   ^X00000002
                          00000002  00C0  1906          .LONG   ^X00000002
                          00000000  00C4  1907          .LONG   ^X00000000
                          00000002  00C8  1908          .LONG   ^X00000002
                          00000002  00CC  1909 2$:      .LONG   ^X00000002                      ;MASK FOR CMPREGMSK PSEUDO OP
                          00000000  00D0  1910 3$:      .LONG   ^X00000000                      ;LOOP COUNT STORAGE
      0B 0A 09 08 07 06 04 03 01 0C  00D4  1911 4$:      .BYTE   12,01,03,04,06,07,08,09,10,11   ;RSRC LDFIELD TABLE
            0F 0D 0E 06 0D 05 05     00DE  1912          .BYTE   05,05,13,06,14,13,15
                                     00E5  1913
      01 00 02 02 01 02 00 01 02 00  00E5  1914 5$:      .BYTE   00,02,01,00,02,01,02,02,00,01   ;SIZE LDFIELD TABLE
            02 00 00 02 02 01 00     00EF  1915          .BYTE   00,01,02,02,00,00,02
                                     00F6  1916
      0C 0A 06 06 07 06 05 03 02 0C  00F6  1917 6$:      .BYTE   12,02,03,05,06,07,06,06,10,12   ;RSRC LDFIELD TABLE
            0F 05 15 0C 0C 0C 05     0100  1918          .BYTE   05,12,12,12,21,05,15
                                     0107  1919 7$:
U 09, 7700,0800,5BE5,0300,0470,180A  0107  1920 ;* 09    Q_R[TEMP0]                              ;U-CODE TO MODIFY EACH
U 0A, 7700,4800,5BF4,034C,0070,180B  0112  1924 ;* 0A    PSL_R[R3]                               ;LOOP
                                     011D  1928 8$:
U 0B, 7701,4800,0390,0001,0470,180C  011D  1929 ;* 0B    SIZE[BYTE],ALU/A+B+CI,MUX/R.Q,RSRC/TEMP0,CCOP2
                                     0128  1933
                                     0128  1934
                                     0128  1935 BEGIN_CPU_DATA
                                     0002  1936
                                     0002  1937 DCS_DATA
                                     0001  1938
U 00, 7700,C800,0364,0300,0470,1801  0001  1939 ;* 00    NOP
U 01, 7700,0800,5BE4,0340,2470,1802  000C  1943 ;* 01    WCTRL/LOADCRAR,WB_R[R0]                 ;DISABLE ^P INTERRUPTS
U 02, 7700,4800,5BE4,0344,2C70,1803  0017  1947 ;* 02    WCTRL/CONWRITE,WB_R[R1]                 ;...
U 03, 7700,0800,5BF4,0348,0070,1804  0022  1951 ;* 03    PSL_R[R2]                               ;IPL = IF NATIVE MODE
U 04, 5700,0840,0364,0350,0470,1805  002D  1955 ;* 04    R[R4]_RDM                               ;LOOP COUNT TO RTEMPS
U 05, 7700,D800,EF64,0301,0470,1806  0038  1959 ;* 05    PL_[2]                                  ;PLATCH GETS 2
U 06, 7700,8800,8B71,0350,0470,1807  0043  1963 ;* 06    Q_R[R4].RL.P                            ;FORM CACHE ADDRESS
U 07, 7700,4800,03A4,03D8,4870,1808  004E  1967 ;* 07    WCTRL/PC_WB,WB_Q                        ;FORCE PREFETCH
U 08, 7700,4800,0364,1700,0470,1809  0059  1971 ;* 08    IRD1TEST                                ;LOAD INST. REGISTER
U 09, 7700,0800,5BE5,0300,0470,180A  0064  1975 ;* 09    Q_R[TEMP0]                              ;TEST PATTERN TO Q REGISTER
U 0A, 7700,4800,5BF4,034C,0070,180B  006F  1979 ;* 0A    PSL_R[R3]                               ;IPL = 1F COMPATABILITY MODE
U 0B, 7701,4800,0390,0001,0470,180C  007A  1983 ;* 0B    SIZE[BYTE],ALU/A+B+CI,MUX/R.Q,RSRC/TEMP0,CCOP2;SET COND. CODES
U 0C, 6700,C800,0364,0300,0870,180D  0085  1987 ;* 0C    RDM_WB,WB_PSL                           ;READ PSL TO RDM
U 0D, 7300,C800,0364,0300,0470,180E  0090  1991 ;* 0D    NOP,STOP.CLOCK                          ;STOP THE CPU CLOCK
                                     009B  1995
                                     009B  1996 RTEMP_DATA
                                     0001  1997
                          40004000   0001  1998          .LONG   ^X40004000                      ;TEST PATTERNS
                          00404040   0005  1999          .LONG   ^X00404040
                          00004040   0009  2000          .LONG   ^X00004040
                          0000C040   000D  2001          .LONG   ^X0000C040
                          40000040   0011  2002          .LONG   ^X40000040
                          00000040   0015  2003          .LONG   ^X00000040
                          80000000   0019  2004          .LONG   ^X80000000
                          00008000   001D  2005          .LONG   ^X00008000
```

```
80400000  0021  2006          .LONG   ^X80400000
80004000  0025  2007          .LONG   ^X80004000
00000080  0029  2008          .LONG   ^X00000080
40408040  002D  2009          .LONG   ^X40408040
00000000  0031  2010          .LONG   ^X00000000
40404040  0035  2011          .LONG   ^X40404040
80000080  0039  2012          .LONG   ^X80000080
00008080  003D  2013          .LONG   ^X00008080
00C00000  0041  2014          .LONG   ^X00C00000                      ;DISABLE ^P INTERRUPTS
00400000  0045  2015          .LONG   ^X00400000                      ;...
001F0000  0049  2016          .LONG   ^X001F0000                      ;IPL = 1F NATIVE MODE
801F0000  004D  2017          .LONG   ^X801F0000                      ;IPL = 1F COMPATABILITY MODE
00000000  0051  2018          .LONG   ^X00000000                      ;LOOP COUNT STORAGE
00004080  0055  2019          .LONG   ^X00004080                      ;TEST PATTERN
          0059  2020
          0059  2021 CACHE_DATA
          0001  2022
00000024  0001  2023          .LONG   ^X00000024                      ;INSTRUCTION TABLE
0000006C  0005  2024          .LONG   ^X0000006C
0000000C  0009  2025          .LONG   ^X0000000C
00000084  000D  2026          .LONG   ^X00000084
000000E4  0011  2027          .LONG   ^X000000E4
00000000  0015  2028          .LONG   ^X00000000
00000064  0019  2029          .LONG   ^X00000064
00000024  001D  2030          .LONG   ^X00000024
000000EA  0021  2031          .LONG   ^X000000EA
00000084  0025  2032          .LONG   ^X00000084                      ;LOOP 10
00000026  0029  2033          .LONG   ^X00000026
00000064  002D  2034          .LONG   ^X00000064
0000004C  0031  2035          .LONG   ^X0000004C
00000084  0035  2036          .LONG   ^X00000084
00000084  0039  2037          .LONG   ^X00000084
00000028  003D  2038          .LONG   ^X00000028
00000024  0041  2039          .LONG   ^X00000024
00000000  0045  2040          .LONG   ^X00000000
          0049  2041
          0049  2042 END_CPU_DATA
          0128  2043
          0128  2044
          0128  2045 END_TEST_DATA
          0128
          0128        ;-----------------------------------------------------------------------------
          0128  2046 ENDTEST
```

```
0011          .SBTTL  "TEST   097     C BIT TEST 1
0011   2048 ;*******************************************************************************
0011   2049 ;++
0011   2050 ;
0011   2051 ; FUNCTIONAL DESCRIPTION:
0011   2052 ;
0011   2053 ;       THIS TEST VERIFIES THAT THE C BIT IN THE CCC GATE ARRAY SETS PROPERLY
0011   2054 ;       FOR NATIVE MODE INSTRUCTIONS USING CC/CCOP1.CCBR_SIGND MICRO ORDER.
0011   2055 ;
0011   2056 ;
0011   2057 ; TEST ALGORITHM:
0011   2058 ;
0011   2059 ;       1.  CREATE A TEST LOOP OF 32
0011   2060 ;       2.  MODIFY THE MICRO CODE FOR EACH LOOP THROUGH THE TEST
0011   2061 ;       3.  LOAD THE NEW MICRO CODE INTO THE RDM DCS
0011   2062 ;       4.  SAVE THE LOOP COUNT
0011   2063 ;       5.  MOVE THE LOOP COUNT TO THE RDM WDREG
0011   2064 ;       6.  EXECUTE THE DCS PROGRAM
0011   2065 ;               a.  DISABLE ^P INTERRUPTS
0011   2066 ;               b.  SET THE IPL TO 1F AND MODE TO NATIVE
0011   2067 ;               c.  MOVE THE LOOP COUNT TO THE RTEMPS
0011   2068 ;               d.  LOAD THE PLATCH WITH 2
0011   2069 ;               e.  ROTATE THE LOOP COUNT LEFT BY 2 TO FORM A CACHE ADDRESS
0011   2070 ;               f.  LOAD THE PC WITH THE CACHE ADDRESS TO FORCE PREFETCH
0011   2071 ;               g.  PERFORM IRD1TEST TO LOAD THE IR REGISTER
0011   2072 ;               h.  LOAD THE Q REGISTER WITH TEST DATA
0011   2073 ;               i.  ADD THE Q REGISTER TO ONE OF THE RTEMPS TO CREATE THE
0011   2074 ;                       TEST CONDITIONS WHILE PERFORMING CCOP1.
0011   2075 ;               j.  READ THE PSL TO THE RDM
0011   2076 ;       7.  TEST THE PSL C BIT FOR CORRECT STATE
0011   2077 ;       8.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST PATTERNS
0011   2078 ;
0011   2079 ;
0011   2080 ; TEST PATTERNS:
0011   2081 ;
0011   2082 ;       ITERATION       IR      Q REG. + R[TEMP*]       SIZE
0011   2083 ;
0011   2084 ;           1           83      00000000 + 00000000     BYTE
0011   2085 ;           2           F0      00008000 + 00008000     WORD
0011   2086 ;           3           C0      80000000 + 80000000     LONG
0011   2087 ;           4           9E      00000000 + 00000000     WORD
0011   2088 ;           5           E4      00000000 + 00000000     LONG
0011   2089 ;           6           90      00000080 + 00000080     BYTE
0011   2090 ;           7           59      00000000 + 00000000     BYTE
0011   2091 ;           8           99      00000000 + 00000000     BYTE
0011   2092 ;           9           D0      00000080 + 00000080     BYTE
0011   2093 ;           A           90      00000000 + 00000000     BYTE
0011   2094 ;           B           00      00000000 + 00000000     WORD
0011   2095 ;           C           6E      00000000 + 00000000     BYTE
0011   2096 ;           D           96      00000000 + 00000000     BYTE
0011   2097 ;           E           B7      00000000 + 00000000     BYTE
0011   2098 ;           F           40      00000080 + 00000080     BYTE
0011   2099 ;          10           8F      00000000 + 00000000     BYTE
0011   2100 ;          11           D7      00000000 + 00000000     BYTE
0011   2101 ;          12           1F      00000000 + 00000000     BYTE
```

```
0011  2102 ;          13          91      00000080 + 00000080     BYTE
0011  2103 ;          14          CD      00000000 + 00000000     BYTE
0011  2104 ;          15          FB      00000000 + 00000000     BYTE
0011  2105 ;          16          51      00000000 + 00000000     BYTE
0011  2106 ;          17          84      00000080 + 00000080     BYTE
0011  2107 ;          18          98      00000080 + 00000080     BYTE
0011  2108 ;          19          00      00008000 + 00008000     WORD
0011  2109 ;          1A          08      00000080 + 00000080     BYTE
0011  2110 ;          1B          10      00000080 + 00000080     BYTE
0011  2111 ;          1C          58      00000000 + 00000000     BYTE
0011  2112 ;          1D          A4      00000000 + 00000000     BYTE
0011  2113 ;          1E          D1      00000080 + 00000080     BYTE
0011  2114 ;          1F          E0      00000000 + 00000000     BYTE
0011  2115 ;          20          F4      00000000 + 00000000     BYTE
0011  2116 ;
0011  2117 ;
0011  2118 ; TEST ALGORITHM:
0011  2119 ;
0011  2120 ;       THESE TEST PATTERNS WERE GENERATED USING A LOGIC SIMULATOR.  SO IF
0011  2121 ;       YOU ATTEMPT TO READ THE VAX SRM AND TRY AN CORRELATE THE INSTRUCTIONS
0011  2122 ;       USED VERSUS CONDITIONS CODES IT REALLY WON'T MAKE ANY SENSE.  THE
0011  2123 ;       ATTEMPT HERE IS TO TEST THE INTERNAL WORKINGS OF THE CCC GATE ARRAY.
0011  2124 ;       ANY FAILUES SHOULD BE CONFINED TO THIS ARRAY OR THE INSTRUCTION
0011  2125 ;       REGISTER FEEDING IT.
0011  2126 ;
0011  2127 ; ASSUMPTIONS:
0011  2128 ;
0011  2129 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
0011  2130 ;
0011  2131 ; ERROR DESCRIPTION:
0011  2132 ;
0011  2133 ;       EXPECTED C BIT STATE
0011  2134 ;       RECEIVED C BIT STATE
0011  2135 ;       LOOP COUNT I
0011  2136 ;
0011  2137 ;--
0011  2138 ;*******************************************************************************
0011  2139
0011  2140
0011  2141          INITIALIZE                                  ;INT THE CPU
0013  2142          LOOP            I,1,32                      ;CREATE A TEST LOOP OF 32
001A  2143          LDFIELD         4$,I,B,7$,34,39,            ;MODIFY U-CODE FOR THIS LOOP
002B  2144          LDFIELD         5$,I,B,8$,40,41,            ;...
003C  2145          LDFIELD         6$,I,B,8$,34,39,            ;...
004D  2146          LOADDCS         7$,,<^X09>,2                ;LOAD U-CODE INTO DCS
0054  2147          SAVEINDEX       I,3$                        ;SAVE LOOP COUNT
0059  2148          LOADREG         WDREG,3$,,L                 ;PUT LOOP COUNT INTO WDREG
0061  2149          ERRLOOP                                     ;ERROR LOOP POINT
0063  2150          FETCH           0                           ;START DCS PROGRAM
0068  2151          BRSTCLK         <^X0C>                      ;END DCS PROGRAM
006C  2152          CMPREGMSK       WHREG,1$,I,2$,,L,           ;TEST C BIT
0078  2153          IFERROR         ,<CCC>,                     ;BIT INCORRECT
007E  2154          ENDLOOP         I                           ;END TEST LOOP
0081  2155          SKIP                                        ;GO TO NEXT TEST
0088  2156
```

```
                        0088  2157
                        0088  2158 BEGIN_TEST_DATA
                        0088
                        0088       ;-----------------------------------------------------------------------
                        0088  2159
                00000000  0088  2160 1$:     .LONG   ^X00000000                      ;RESULTS TABLE
                00000000  008C  2161         .LONG   ^X00000000
                00000001  0090  2162         .LONG   ^X00000001
                00000001  0094  2163         .LONG   ^X00000001
                00000001  0098  2164         .LONG   ^X00000001
                00000001  009C  2165         .LONG   ^X00000001
                00000001  00A0  2166         .LONG   ^X00000001
                00000000  00A4  2167         .LONG   ^X00000000
                00000001  00A8  2168         .LONG   ^X00000001
                00000001  00AC  2169         .LONG   ^X00000001                      ;LOOP 10
                00000001  00B0  2170         .LONG   ^X00000001
                00000000  00B4  2171         .LONG   ^X00000000
                00000000  00B8  2172         .LONG   ^X00000000
                00000001  00BC  2173         .LONG   ^X00000001
                00000000  00C0  2174         .LONG   ^X00000000
                00000001  00C4  2175         .LONG   ^X00000001
                00000001  00C8  2176         .LONG   ^X00000001
                00000000  00CC  2177         .LONG   ^X00000000
                00000000  00D0  2178         .LONG   ^X00000000
                00000000  00D4  2179         .LONG   ^X00000000                      ;LOOP 20
                00000000  00D8  2180         .LONG   ^X00000000
                00000000  00DC  2181         .LONG   ^X00000000
                00000000  00E0  2182         .LONG   ^X00000000
                00000000  00E4  2183         .LONG   ^X00000000
                00000001  00E8  2184         .LONG   ^X00000001
                00000000  00EC  2185         .LONG   ^X00000000
                00000000  00F0  2186         .LONG   ^X00000000
                00000000  00F4  2187         .LONG   ^X00000000
                00000000  00F8  2188         .LONG   ^X00000000
                00000000  00FC  2189         .LONG   ^X00000000                      ;LOOP 30
                00000000  0100  2190         .LONG   ^X00000000
                00000000  0104  2191         .LONG   ^X00000000
                00000001  0108  2192 2$:     .LONG   ^X00000001                      ;MASK FOR CMPREGMSK PSEUDO OP
                00000000  010C  2193 3$:     .LONG   ^X00000000                      ;LOOP COUNT STORAGE
      00 03 00 00 03 00 00 02 01 00  0110  2194 4$:     .BYTE   00,01,02,00,00,03,00,00,03,00   ;RSRC LDFIELD TABLE
      00 03 00 00 00 03 00 00 00 00  011A  2195         .BYTE   00,00,00,00,03,00,00,00,03,00
      03 00 00 03 03 01 03 03 00 00  0124  2196         .BYTE   00,00,03,03,01,03,03,00,00,03
                                 00 00  012E  2197         .BYTE   00,00
                                        0130  2198
      00 00 00 00 00 02 01 02 01 00  0130  2199 5$:     .BYTE   00,01,02,01,02,00,00,00,00,00   ;SIZE LDFIELD TABLE
      00 00 00 00 00 00 00 00 00 01  013A  2200         .BYTE   01,00,00,00,00,00,00,00,00,00
      00 00 00 00 00 01 00 00 00 00  0144  2201         .BYTE   00,00,00,00,01,00,00,00,00,00
                                 00 00  014E  2202         .BYTE   00,00
                                        0150  2203
      00 03 00 00 03 00 00 02 01 00  0150  2204 6$:     .BYTE   00,01,02,00,00,03,00,00,03,00   ;RSRC LDFIELD TABLE
      00 03 00 00 00 03 00 00 00 00  015A  2205         .BYTE   00,00,00,00,03,00,00,00,03,00
      03 00 00 03 03 01 03 03 00 00  0164  2206         .BYTE   00,00,03,03,01,03,03,00,00,03
                                 00 00  016E  2207         .BYTE   00,00
                                        0170  2208 7$:
U 09, 7700,0800,5BE5,0300,0470,180A     0170  2209 ;% 09    Q_R[TEMP0]                      ;U-CODE TO MODIFY EACH
```

```
                                    017B  8$:                                                  ;LOOP
U 0A, 7701,C800,0390,0000,8470,180B 017B  ;% 0A   SIZE[BYTE],ALU/A+B+CI,MUX/R.Q,RSRC/TEMP0,CCOP1
                                    0186
                                    0186
                                    0186  BEGIN_CPU_DATA
                                    0002
                                    0002  DCS_DATA
                                    0001
U 00, 7700,C800,0364,0300,0470,1801 0001  ;% 00   NOP
U 01, 7700,0800,5BE4,0310,2470,1802 000C  ;% 01   WCTRL/LOADCRAR,WB_R[TEMP4]                  ;DISABLE ^P INTERRUPTS
U 02, 7700,4800,5BE4,0314,2C70,1803 0017  ;% 02   WCTRL/CONWRITE,WB_R[TEMP5]                  ;...
U 03, 7700,0800,5BF4,0318,0070,1804 0022  ;% 03   PSL_R[TEMP6]                                ;IPL = 1F NATIVE MODE
U 04, 5700,4840,0364,031C,0470,1805 002D  ;% 04   R[TEMP7]_RDM                                ;LOOP COUNT TO RTEMPS
U 05, 7700,D800,EF64,0301,0470,1806 0038  ;% 05   PL_[2]                                      ;PLATCH GETS 2
U 06, 7700,C800,8B71,031C,0470,1807 0043  ;% 06   Q_R[TEMP7].RL.P                             ;FORM CACHE ADDRESS
U 07, 7700,4800,03A4,03D8,4870,1808 004E  ;% 07   WCTRL/PC_WB,WB_Q                            ;FORCE PREFETCH
U 08, 7700,4800,0364,1700,0470,1809 0059  ;% 08   IRD1TEST                                    ;LOAD INST. REGISTER
U 09, 7700,0800,5BE5,0300,0470,180A 0064  ;% 09   Q_R[TEMP0]                                  ;Q GETS TEST PATTERN
U 0A, 7701,C800,0390,0000,8470,180B 006F  ;% 0A   SIZE[BYTE],ALU/A+B+CI,MUX/R.Q,RSRC/TEMP0,CCOP1;SET COND CODES
U 0B, 6700,4800,0364,0300,0870,180C 007A  ;% 0B   RDM_WB,WB_PSL                               ;READ PSL TO RDM
U 0C, 7300,C800,0364,0300,0470,180D 0085  ;% 0C   NOP,STOP.CLOCK                              ;STOP THE CPU CLOCK
                                    0090
                                    0090  RTEMP_DATA
                                    0001
                         00000000   0001          .LONG   ^X00000000                          ;TEST PATTERNS
                         00008000   0005          .LONG   ^X00008000
                         80000000   0009          .LONG   ^X80000000
                         00000080   000D          .LONG   ^X00000080
                         00C00000   0011          .LONG   ^X00C00000                          ;DISABLE ^P INTERRUPTS
                         00400000   0015          .LONG   ^X00400000                          ;...
                         001F0000   0019          .LONG   ^X001F0000                          ;IPL = 1F NATIVE.MODE
                         00000000   001D          .LONG   ^X00000000                          ;LOOP COUNT STORAGE
                                    0021
                                    0021  CACHE_DATA
                                    0001
                         00000083   0001          .LONG   ^X00000083                          ;INSTRUCTION TABLE
                         000000F0   0005          .LONG   ^X000000F0
                         000000C0   0009          .LONG   ^X000000C0
                         0000009E   000D          .LONG   ^X0000009E
                         000000E4   0011          .LONG   ^X000000E4
                         00000090   0015          .LONG   ^X00000090
                         00000059   0019          .LONG   ^X00000059
                         00000099   001D          .LONG   ^X00000099
                         000000D0   0021          .LONG   ^X000000D0
                         00000090   0025          .LONG   ^X00000090                          ;LOOP 10
                         00000000   0029          .LONG   ^X00000000
                         0000006E   002D          .LONG   ^X0000006E
                         00000096   0031          .LONG   ^X00000096
                         000000B7   0035          .LONG   ^X000000B7
                         00000040   0039          .LONG   ^X00000040
                         0000008F   003D          .LONG   ^X0000008F
                         000000D7   0041          .LONG   ^X000000D7
                         0000001F   0045          .LONG   ^X0000001F
                         00000091   0049          .LONG   ^X00000091
                         000000CD   004D          .LONG   ^X000000CD                          ;LOOP 20
```

```
000000FB  0051  2310          .LONG   ^X000000FB
00000051  0055  2311          .LONG   ^X00000051
00000084  0059  2312          .LONG   ^X00000084
00000098  005D  2313          .LONG   ^X00000098
00000000  0061  2314          .LONG   ^X00000000
00000008  0065  2315          .LONG   ^X00000008
00000010  0069  2316          .LONG   ^X00000010
00000058  006D  2317          .LONG   ^X00000058
000000A4  0071  2318          .LONG   ^X000000A4
000000D1  0075  2319          .LONG   ^X000000D1                    ;LOOP 30
000000E0  0079  2320          .LONG   ^X000000E0
000000F4  007D  2321          .LONG   ^X000000F4
00000000  0081  2322          .LONG   ^X00000000
          0085  2323
          0085  2324 END_CPU_DATA
          0186  2325
          0186  2326
          0186  2327 END_TEST_DATA
          0186
          0186        ;-------------------- ----------------------------------------------------
          0186  2328 ENDTEST
```

```
                        0011        .SBTTL  "TEST   098     C BIT TEST 2
                        0011  2330 ;*******************************************************************************
                        0011  2331 ;++
                        0011  2332 ;
                        0011  2333 ; FUNCTIONAL DESCRIPTION:
                        0011  2334 ;
                        0011  2335 ;       THIS TEST VERIFIES THAT THE C BIT IN THE CCC GATE ARRAY SETS PROPERLY
                        0011  2336 ;       FOR COMPATABILITY MODE INSTRUCTIONS USING CC/CCOP1.CCBR_SIGND
                        0011  2337 ;       MICRO ORDER.
                        0011  2338 ;
                        0011  2339 ;
                        0011  2340 ; TEST ALGORITHM:
                        0011  2341 ;
                        0011  2342 ;       1.  CREATE A TEST LOOP OF 12
                        0011  2343 ;       2.  MODIFY THE MICRO CODE FOR EACH LOOP THROUGH THE TEST
                        0011  2344 ;       3.  LOAD THE NEW MICRO CODE INTO THE RDM DCS
                        0011  2345 ;       4.  SAVE THE LOOP COUNT
                        0011  2346 ;       5.  MOVE THE LOOP COUNT TO THE RDM WDREG
                        0011  2347 ;       6.  EXECUTE THE DCS PROGRAM
                        0011  2348 ;               a.  DISABLE ^P INTERRUPTS
                        0011  2349 ;               b.  SET THE IPL TO 1F AND MODE TO NATIVE
                        0011  2350 ;               c.  MOVE THE LOOP COUNT TO THE RTEMPS
                        0011  2351 ;               d.  LOAD THE PLATCH WITH 2
                        0011  2352 ;               e.  ROTATE THE LOOP COUNT LEFT BY 2 TO FORM A CACHE ADDRESS
                        0011  2353 ;               f.  LOAD THE PC WITH THE CACHE ADDRESS TO FORCE PREFETCH
                        0011  2354 ;               g.  PERFORM IRD1TEST TO LOAD THE IR REGISTER
                        0011  2355 ;               h.  LOAD THE Q REGISTER WITH TEST DATA
                        0011  2356 ;               i.  SET IPL TO 1F AND COMPATABILITY MODE
                        0011  2357 ;               j.  ADD THE Q REGISTER TO ONE OF THE RTEMPS TO CREATE THE
                        0011  2358 ;                       TEST CONDITIONS WHILE PERFORMING CCOP1.
                        0011  2359 ;               k.  READ THE PSL TO THE RDM
                        0011  2360 ;       7.  TEST THE PSL C BIT FOR CORRECT STATE
                        0011  2361 ;       8.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST PATTERNS
                        0011  2362 ;
                        0011  2363 ;
                        0011  2364 ; TEST PATTERNS:
                        0011  2365 ;
                        0011  2366 ;       ITERATION       IR      Q REG. + R[TEMP*]       SIZE
                        0011  2367 ;
                        0011  2368 ;           1           22      00008000 + 00008000     WORD
                        0011  2369 ;           2           A6      80000080 + 00000080     BYTE
                        0011  2370 ;           3           A0      80000000 + 80000000     LONG
                        0011  2371 ;           4           46      80000000 + 00000000     BYTE
                        0011  2372 ;           5           04      80000000 + 00000000     BYTE
                        0011  2373 ;           6           06      00000000 + 00000000     WORD
                        0011  2374 ;           7           06      00000000 + 00000000     LONG
                        0011  2375 ;           8           06      00000080 + 00000080     BYTE
                        0011  2376 ;           9           B3      00000080 + 00000080     BYTE
                        0011  2377 ;           A           20      00000080 + 00000080     BYTE
                        0011  2378 ;           B           82      80000080 + 00008000     WORD
                        0011  2379 ;           C           04      00000080 + 00000080     BYTE
                        0011  2380 ;
                        0011  2381 ;
                        0011  2382 ; TEST ALGORITHM:
                        0011  2383 ;
```

```
                     0011  2384 ;       THESE TEST PATTERNS WERE GENERATED USING A LOGIC SIMULATOR.  SO IF
                     0011  2385 ;       YOU ATTEMPT TO READ THE VAX SRM AND TRY AN CORRELATE THE INSTRUCTIONS
                     0011  2386 ;       USED VERSUS CONDITIONS CODES IT REALLY WON'T MAKE ANY SENSE.  THE
                     0011  2387 ;       ATTEMPT HERE IS TO TEST THE INTERNAL WORKINGS OF THE CCC GATE ARRAY.
                     0011  2388 ;       ANY FAILUES SHOULD BE CONFINED TO THIS ARRAY OR THE INSTRUCTION
                     0011  2389 ;       REGISTER FEEDING IT.
                     0011  2390 ;
                     0011  2391 ; ASSUMPTIONS:
                     0011  2392 ;
                     0011  2393 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
                     0011  2394 ;
                     0011  2395 ; ERROR DESCRIPTION:
                     0011  2396 ;
                     0011  2397 ;       EXPECTED C BIT STATE
                     0011  2398 ;       RECEIVED C BIT STATE
                     0011  2399 ;       LOOP COUNT I
                     0011  2400 ;
                     0011  2401 ;--
                     0011  2402 ;*******************************************************************************
                     0011  2403
                     0011  2404
                     0011  2405         INITIALIZE                                  ;INT THE CPU
                     0013  2406         LOOP            I,1,12                      ;CREATE A TEST LOOP OF 12
                     001A  2407         LDFIELD         4$,I,B,7$,34,39,            ;MODIFY U-CODE FOR THIS LOOP
                     002B  2408         LDFIELD         5$,I,B,8$,40,41,            ;...
                     003C  2409         LDFIELD         6$,I,B,8$,34,39,            ;...
                     004D  2410         LOADDCS         7$,,<^X09>,3                ;LOAD U-CODE INTO DCS
                     0054  2411         SAVEINDEX       I,3$                        ;SAVE LOOP COUNT
                     0059  2412         LOADREG         WDREG,3$,,L                 ;PUT LOOP COUNT INTO WDREG
                     0061  2413         ERRLOOP                                     ;ERROR LOOP POINT
                     0063  2414         FETCH           0                           ;START DCS PROGRAM
                     0068  2415         BRSTCLK         <^X0D>                      ;END DCS PROGRAM
                     006C  2416         CMPREGMSK       WHREG,1$,I,2$,,L,           ;TEST C BIT
                     0078  2417         IFERROR         ,<CCC>,                     ;BIT INCORRECT
                     007E  2418         ENDLOOP         I                           ;END TEST LOOP
                     0081  2419         SKIP                                        ;GO TO NEXT TEST
                     0088  2420
                     0088  2421
                     0088  2422 BEGIN_TEST_DATA
                     0088
                     0088         ;-------------------------------------------------------------------------
                     0088  2423
00000001             0088  2424 1$:      .LONG   ^X00000001                          ;RESULTS TABLE
00000000             008C  2425         .LONG   ^X00000000
00000001             0090  2426         .LONG   ^X00000001
00000000             0094  2427         .LONG   ^X00000000
00000001             0098  2428         .LONG   ^X00000001
00000000             009C  2429         .LONG   ^X00000000
00000000             00A0  2430         .LONG   ^X00000000
00000001             00A4  2431         .LONG   ^X00000001
00000000             00A8  2432         .LONG   ^X00000000
00000000             00AC  2433         .LONG   ^X00000000                          ;LOOP 10
00000000             00B0  2434         .LONG   ^X00000000
00000000             00B4  2435         .LONG   ^X00000000
00000001             00B8  2436 2$:      .LONG   ^X00000001                          ;MASK FOR CMPREGMSK PSEUDO OP
```

```
                          00000000  00BC  2437 3$:      .LONG   ^X00000000                        ;LOOP COUNT STORAGE
      04 04 04 03 03 02 02 02 01 00  00C0  2438 4$:      .BYTE   00,01,02,02,02,03,03,04,04,04     ;RSRC LDFIELD TABLE
                              04 01  00CA  2439          .BYTE   01,04
                                     00CC  2440
      00 00 00 02 01 00 00 02 00 01  00CC  2441 5$:      .BYTE   01,00,02,00,00,01,02,00,00,00     ;SIZE LDFIELD TABLE
                              00 00  00D6  2442          .BYTE   00,00
                                     00D8  2443
      04 04 04 03 03 03 03 02 04 00  00D8  2444 6$:      .BYTE   00,04,02,03,03,03,03,04,04,04     ;RSRC LDFIELD TABLE
                              04 04  00E2  2445          .BYTE   04,04
                                     00E4  2446 7$:
U 09, 7700,0800,5BE5,0300,0470,180A  00E4  2447 ;% 09   Q_R[TEMP0]                                ;U-CODE TO MODIFY EACH
U 0A, 7700,4800,5BF4,0320,0070,180B  00EF  2451 ;% 0A   PSL_R[TEMP8]                              ;LOOP
                                     00FA  2455 8$:
U 0B, 7701,0800,0390,0100,8470,180C  00FA  2456 ;% 0B   SIZE[WORD],ALU/A+B+CI,MUX/R.Q,RSRC/TEMP0,CCOP1
                                     0105  2460
                                     0105  2461
                                     0105  2462 BEGIN_CPU_DATA
                                     0002  2463
                                     0002  2464 DCS_DATA
                                     0001  2465
U 00, 7700,C800,0364,0300,0470,1801  0001  2466 ;% 00   NOP
U 01, 7700,4800,5BE4,0314,2470,1802  000C  2470 ;% 01   WCTRL/LOADCRAR,WB_R[TEMP5]                ;DISABLE ^P INTERRUPTS
U 02, 7700,4800,5BE4,0318,2C70,1803  0017  2474 ;% 02   WCTRL/CONWRITE,WB_R[TEMP6]                ;...
U 03, 7700,4800,5BF4,031C,0070,1804  0022  2478 ;% 03   PSL_R[TEMP7]                              ;IPL = 1F NATIVE MODE
U 04, 5700,0840,0364,0324,0470,1805  002D  2482 ;% 04   R[TEMP9]_RDM                              ;LOOP COUNT TO RTEMPS
U 05, 7700,D800,EF64,0301,0470,1806  0038  2486 ;% 05   PL_[2]                                    ;PLATCH GETS 2
U 06, 7700,8800,8B71,0324,0470,1807  0043  2490 ;% 06   Q_R[TEMP9].RL.P                           ;FORM CACHE ADDRESS
U 07, 7700,4800,03A4,03D8,4870,1808  004E  2494 ;% 07   WCTRL/PC_WB,WB_Q                          ;FORCE PREFETCH
U 08, 7700,4800,0364,1700,0470,1809  0059  2498 ;% 08   IRD1TEST                                  ;LOAD INST. REGISTER
U 09, 7700,0800,5BE5,0300,0470,180A  0064  2502 ;% 09   Q_R[TEMP0]                                ;TEST PATTERN TO Q REGISTER
U 0A, 7700,4800,5BF4,0320,0070,180B  006F  2506 ;% 0A   PSL_R[TEMP8]                              ;IPL = 1F COMPATABILITY MODE
U 0B, 7701,0800,0390,0100,8470,180C  007A  2510 ;% 0B   SIZE[WORD],ALU/A+B+CI,MUX/R.Q,RSRC/TEMP0,CCOP1;SET COND. CODES
U 0C, 6700,C800,0364,0300,0870,180D  0085  2514 ;% 0C   RDM_WB,WB_PSL                             ;READ PSL TO RDM
U 0D, 7300,C800,0364,0300,0470,180E  0090  2518 ;% 0D   NOP,STOP.CLOCK                            ;STOP THE CPU CLOCK
                                     009B  2522
                                     009B  2523 RTEMP_DATA
                                     0001  2524
                          00008000  0001  2525          .LONG   ^X00008000                        ;TEST PATTERNS
                          80000080  0005  2526          .LONG   ^X80000080
                          80000000  0009  2527          .LONG   ^X80000000
                          00000000  000D  2528          .LONG   ^X00000000
                          00000080  0011  2529          .LONG   ^X00000080
                          00C00000  0015  2530          .LONG   ^X00C00000                        ;DISABLE ^P INTERRUPTS
                          00400000  0019  2531          .LONG   ^X00400000                        ;...
                          001F0000  001D  2532          .LONG   ^X001F0000                        ;IPL = 1F NATIVE MODE
                          801F0000  0021  2533          .LONG   ^X801F0000                        ;IPL = 1F COMPATABILITY MODE
                          00000000  0025  2534          .LONG   ^X00000000                        ;LOOP COUNT STORAGE
                                     0029  2535
                                     0029  2536 CACHE_DATA
                                     0001  2537
                          00000022  0001  2538          .LONG   ^X00000022                        ;INSTRUCTION TABLE
                          000000A6  0005  2539          .LONG   ^X000000A6
                          000000A0  0009  2540          .LONG   ^X000000A0
                          00000046  000D  2541          .LONG   ^X00000046
                          00000004  0011  2542          .LONG   ^X00000004
```

```
00000006  0015  2543          .LONG   ^X00000006
00000006  0019  2544          .LONG   ^X00000006
00000006  001D  2545          .LONG   ^X00000006
000000B3  0021  2546          .LONG   ^X000000B3
00000020  0025  2547          .LONG   ^X00000020                      ;LOOP 10
00000082  0029  2548          .LONG   ^X00000082
00000004  002D  2549          .LONG   ^X00000004
00000000  0031  2550          .LONG   ^X00000000
          0035  2551
          0035  2552 END_CPU_DATA
          0105  2553
          0105  2554
          0105  2555 END_TEST_DATA
          0105
          0105        ;-------------------------------------------------------------------------
          0105  2556 ENDTEST
```

```
001C          .SBTTL  "TEST   099     BRANCH TEST NATIVE MODE
001C  2558 ;*******************************************************************************
001C  2559 ;++
001C  2560 ;
001C  2561 ; FUNCTIONAL DESCRIPTION:
001C  2562 ;
001C  2563 ;       THIS TEST VERIFIES THAT THE CCBR0 SIGNAL FROM THE CCC GATE ARRAY CAN
001C  2564 ;       BE SET PROPERLY FOR NATIVE MODE BRANCH CONDITIONS.
001C  2565 ;
001C  2566 ;
001C  2567 ; TEST ALGORITHM:
001C  2568 ;
001C  2569 ;       1.  CREATE A TEST LOOP OF 24
001C  2570 ;       2.  MODIFY THE MICRO CODE FOR EACH LOOP THROUGH THE TEST
001C  2571 ;       3.  LOAD THE NEW MICRO CODE INTO THE RDM DCS
001C  2572 ;       4.  SAVE THE LOOP COUNT
001C  2573 ;       5.  MOVE THE LOOP COUNT TO THE RDM WDREG
001C  2574 ;       6.  EXECUTE THE DCS PROGRAM
001C  2575 ;               a.  SET THE PSL TO THE TEST CONDITIONS
001C  2576 ;               b.  DISABLE ^P INTERRUPTS
001C  2577 ;               c.  MOVE THE LOOP COUNT TO THE RTEMPS
001C  2578 ;               d.  LOAD THE PLATCH WITH 2
001C  2579 ;               e.  ROTATE THE LOOP COUNT LEFT BY 2 TO FORM A CACHE ADDRESS
001C  2580 ;               f.  LOAD THE PC WITH THE CACHE ADDRESS TO FORCE PREFETCH
001C  2581 ;               g.  PERFORM IRD1TEST TO LOAD THE IR REGISTER
001C  2582 ;               h.  BUT ON THE BRANCH CONDITONS
001C  2583 ;       7.  TEST THE CS ADDRESS FOR THE CORRECT BRANCH
001C  2584 ;       8.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST PATTERNS
001C  2585 ;
001C  2586 ;
001C  2587 ; TEST PATTERNS:
001C  2588 ;
001C  2589 ;       ITERATION       IR      PSL
001C  2590 ;
001C  2591 ;           1           1D      001F000D
001C  2592 ;           2           1A      001F0000
001C  2593 ;           3           15      001F0004
001C  2594 ;           4           18      001F0008
001C  2595 ;           5           10      001F0004
001C  2596 ;           6           1C      001F0000
001C  2597 ;           7           1B      001F0001
001C  2598 ;           8           17      001F0007
001C  2599 ;           9           1E      001F0000
001C  2600 ;           A           10      001F0008
001C  2601 ;           B           14      001F0000
001C  2602 ;           C           14      001F0004
001C  2603 ;           D           15      001F0008
001C  2604 ;           E           1B      001F0000
001C  2605 ;           F           1C      001F0002
001C  2606 ;          10           1B      001F0004
001C  2607 ;          11           1D      001F0002
001C  2608 ;          12           1E      001F0001
001C  2609 ;          13           15      001F0000
001C  2610 ;          14           1A      001F0004
001C  2611 ;          15           10      001F000C
```

```
                        001C  2612 ;          16          16      001F000A
                        001C  2613 ;          17          1A      001F0001
                        001C  2614 ;          18          19      001F0004
                        001C  2615 ;
                        001C  2616 ;
                        001C  2617 ; TEST ALGORITHM:
                        001C  2618 ;
                        001C  2619 ;       THESE TEST PATTERNS WERE GENERATED USING A LOGIC SIMULATOR.  SO IF
                        001C  2620 ;       YOU ATTEMPT TO READ THE VAX SRM AND TRY AN CORRELATE THE INSTRUCTIONS
                        001C  2621 ;       USED VERSUS BRANCH CONDITIONS IT REALLY WON'T MAKE ANY SENSE.  THE
                        001C  2622 ;       ATTEMPT HERE IS TO TEST THE INTERNAL WORKINGS OF THE CCC GATE ARRAY.
                        001C  2623 ;       ANY FAILUES SHOULD BE CONFINED TO THIS ARRAY OR THE INSTRUCTION
                        001C  2624 ;       REGISTER FEEDING IT.
                        001C  2625 ;
                        001C  2626 ; ASSUMPTIONS:
                        001C  2627 ;
                        001C  2628 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
                        001C  2629 ;
                        001C  2630 ; ERROR DESCRIPTION:
                        001C  2631 ;
                        001C  2632 ;       EXPECTED CS ADDRESS
                        001C  2633 ;       RECEIVED CS ADDRESS
                        001C  2634 ;       LOOP COUNT I
                        001C  2635 ;
                        001C  2636 ;--
                        001C  2637 ;*******************************************************************************
                        001C  2638
                        001C  2639
                        001C  2640          INITIALIZE                                  ;INT THE CPU
                        001E  2641          LOOP            I,1,24                      ;CREATE A TEST LOOP OF 24
                        0025  2642          LDFIELD         4$,I,B,5$,34,39,            ;MODIFY U-CODE FOR THIS LOOP
                        0036  2643          LOADDCS         5$,,01,1                    ;LOAD U-CODE INTO DCS
                        003D  2644          SAVEINDEX       I,3$                        ;SAVE LOOP COUNT
                        0042  2645          LOADREG         WDREG,3$,,L                 ;PUT LOOP COUNT INTO WDREG
                        004A  2646          ERRLOOP                                     ;ERROR LOOP POINT
                        004C  2647          FETCH           0                           ;START DCS PROGRAM
                        0051  2648          BRSTCLK         <^X0B>                      ;END DCS PROGRAM
                        0055  2649          CMPREGMSK       CSTRP,1$,I,2$,,W,           ;TEST CS ADDRESS
                        0061  2650          IFERROR         ,<CCC>,                     ;ADDRESS INCORRECT
                        0067  2651          ENDLOOP         I                           ;END TEST LOOP
                        006A  2652          SKIP                                        ;GO TO NEXT TEST
                        0071  2653
                        0071  2654
                        0071  2655 BEGIN_TEST_DATA
                        0071
                        0071       ;-------------------------------------------------------------------------
                        0071  2656
1800                    0071  2657 1$:       .WORD   ^X1800                          ;RESULTS TABLE
1801                    0073  2658           .WORD   ^X1801
1801                    0075  2659           .WORD   ^X1801
1800                    0077  2660           .WORD   ^X1800
1801                    0079  2661           .WORD   ^X1801
1801                    007B  2662           .WORD   ^X1801
1801                    007D  2663           .WORD   ^X1801
1800                    007F  2664           .WORD   ^X1800
```

```
                                1801  0081  2665          .WORD   ^X1801
                                1801  0083  2666          .WORD   ^X1801                          ;LOOP 10
                                1801  0085  2667          .WORD   ^X1801
                                1800  0087  2668          .WORD   ^X1800
                                1801  0089  2669          .WORD   ^X1801
                                1800  008B  2670          .WORD   ^X1800
                                1800  008D  2671          .WORD   ^X1800
                                1801  008F  2672          .WORD   ^X1801
                                1801  0091  2673          .WORD   ^X1801
                                1800  0093  2674          .WORD   ^X1800
                                1800  0095  2675          .WORD   ^X1800
                                1800  0097  2676          .WORD   ^X1800                          ;LOOP 20
                                1800  0099  2677          .WORD   ^X1800
                                1800  009B  2678          .WORD   ^X1800
                                1800  009D  2679          .WORD   ^X1800
                                1800  009F  2680          .WORD   ^X1800
                                3FFF  00A1  2681 2$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                            00000000  00A3  2682 3$:      .LONG   ^X00000000                      ;LOOP COUNT STORAGE
     01 00 06 04 00 02 01 02 00 05  00A7  2683 4$:      .BYTE   05,00,02,01,02,00,04,06,00,01   ;RSRC LDFIELD TABLE
     02 00 04 03 02 03 00 01 02 00  00B1  2684          .BYTE   00,02,01,00,03,02,03,04,00,02
                   02 04 08 07      00BB  2685          .BYTE   07,08,04,02
                                    00BF  2686
                                    00BF  2687 5$:
U 01, 7700,0800,5BF4,0314,0070,1802 00BF  2688 ;* 01   PSL_R[TEMP5]                             ;U-CODE TO MODIFY
                                    00CA  2692
                                    00CA  2693
                                    00CA  2694
                                    00CA  2695 BEGIN_CPU_DATA
                                    0002  2696
                                    0002  2697 DCS_DATA
                                    0001  2698
U 00, 7700,C800,0364,0300,0470,1801 0001  2699 ;* 00   NOP
U 01, 7700,0800,5BF4,0314,0070,1802 000C  2703 ;* 01   PSL_R[TEMP5]                             ;SET UP PSL CONDITIONS
U 02, 7700,C800,5BE4,0324,2470,1803 0017  2707 ;* 02   WCTRL/LOADCRAR,WB_R[TEMP9]               ;DISABLE ^P INTERRUPTS
U 03, 7700,C800,5BE4,0328,2C70,1804 0022  2711 ;* 03   WCTRL/CONWRITE,WB_R[TEMP10]              ;...
U 04, 5700,4840,0364,032C,0470,1805 002D  2715 ;* 04   R[TEMP11]_RDM                           ;LOOP COUNT TO RTEMPS
U 05, 7700,D800,EF64,0301,0470,1806 0038  2719 ;* 05   PL_[2]                                 ;PLATCH GETS 2
U 06, 7700,C800,8B71,032C,0470,1807 0043  2723 ;* 06   Q_R[TEMP11].RL.P                       ;FORM CACHE ADDRESS
U 07, 7700,4800,03A4,03D8,4870,1808 004E  2727 ;* 07   WCTRL/PC_WB,WB_Q                        ;FORCE PREFETCH
U 08, 7700,4800,0364,1700,0470,1809 0059  2731 ;* 08   IRD1TEST                               ;LOAD INST. REGISTER
U 09, 7700,4800,0364,0300,0470,180A 0064  2735 ;* 09   NOP
U 0A, 7700,4800,0364,B701,8E70,1800 006F  2739 ;* 0A   BUT/CCBR,CCMISC/NOP.CCBR_BRATST,NEXT/1800;BUT ON BRANCH CONDITIONS
U 0B, 7B00,4800,0364,0300,0470,180C 007A  2743 ;* 0B   NOP,STOP.CLOCK,LATCH.CS.ADDR             ;STOP THE CPU CLOCK
                                    0085  2747
                                    0085  2748 RTEMP_DATA
                                    0001  2749
                            001F0000  0001  2750          .LONG   ^X001F0000                    ;TEST PSL'S
                            001F0008  0005  2751          .LONG   ^X001F0008
                            001F0004  0009  2752          .LONG   ^X001F0004
                            001F0002  000D  2753          .LONG   ^X001F0002
                            001F0001  0011  2754          .LONG   ^X001F0001
                            001F000D  0015  2755          .LONG   ^X001F000D
                            001F0007  0019  2756          .LONG   ^X001F0007
                            001F000C  001D  2757          .LONG   ^X001F000C
                            001F000A  0021  2758          .LONG   ^X001F000A
```

```
00C00000  0025  2759          .LONG   ^X00C00000                      ;DISABLE ^P INTERRUPTS
00400000  0029  2760          .LONG   ^X00400000                      ;...
00000000  002D  2761          .LONG   ^X00000000                      ;LOOP COUNT STORAGE
          0031  2762
          0031  2763 CACHE_DATA
          0001  2764
0000001D  0001  2765          .LONG   ^X0000001D                      ;INSTRUCTION TABLE
0000001A  0005  2766          .LONG   ^X0000001A
00000015  0009  2767          .LONG   ^X00000015
00000018  000D  2768          .LONG   ^X00000018
00000010  0011  2769          .LONG   ^X00000010
0000001C  0015  2770          .LONG   ^X0000001C
0000001B  0019  2771          .LONG   ^X0000001B
00000017  001D  2772          .LONG   ^X00000017
0000001E  0021  2773          .LONG   ^X0000001E
00000010  0025  2774          .LONG   ^X00000010                      ;LOOP 10
00000014  0029  2775          .LONG   ^X00000014
00000014  002D  2776          .LONG   ^X00000014
00000015  0031  2777          .LONG   ^X00000015
0000001B  0035  2778          .LONG   ^X0000001B
0000001C  0039  2779          .LONG   ^X0000001C
0000001B  003D  2780          .LONG   ^X0000001B
0000001D  0041  2781          .LONG   ^X0000001D
0000001E  0045  2782          .LONG   ^X0000001E
00000015  0049  2783          .LONG   ^X00000015
0000001A  004D  2784          .LONG   ^X0000001A                      ;LOOP 20
00000010  0051  2785          .LONG   ^X00000010
00000016  0055  2786          .LONG   ^X00000016
0000001A  0059  2787          .LONG   ^X0000001A
00000019  005D  2788          .LONG   ^X00000019
00000000  0061  2789          .LONG   ^X00000000
          0065  2790
          0065  2791 END_CPU_DATA
          00CA  2792
          00CA  2793
          00CA  2794 END_TEST_DATA
          00CA
          00CA          ;-----------------------------------------------------------------------------
          00CA  2795 ENDTEST
```

```
0023        .SBTTL  "TEST   09A     BRANCH TEST COMPATABILITY MODE
0023  2797 ;*******************************************************************************
0023  2798 ;++
0023  2799 ;
0023  2800 ; FUNCTIONAL DESCRIPTION:
0023  2801 ;
0023  2802 ;       THIS TEST VERIFIES THAT THE CCBR0 SIGNAL FROM THE CCC GATE ARRAY CAN
0023  2803 ;       BE SET PROPERLY BY COMPATABILITY MODE BRANCH CONDITIONS.
0023  2804 ;
0023  2805 ;
0023  2806 ; TEST ALGORITHM:
0023  2807 ;
0023  2808 ;       1.  CREATE A TEST LOOP OF 29
0023  2809 ;       2.  MODIFY THE MICRO CODE FOR EACH LOOP THROUGH THE TEST
0023  2810 ;       3.  LOAD THE NEW MICRO CODE INTO THE RDM DCS
0023  2811 ;       4.  SAVE THE LOOP COUNT
0023  2812 ;       5.  MOVE THE LOOP COUNT TO THE RDM WDREG
0023  2813 ;       6.  EXECUTE THE DCS PROGRAM
0023  2814 ;               a.  SET THE PSL TO NATIVE MODE AND IPL EQUAL 1F
0023  2815 ;               b.  DISABLE ^P INTERRUPTS
0023  2816 ;               c.  MOVE THE LOOP COUNT TO THE RTEMPS
0023  2817 ;               d.  LOAD THE PLATCH WITH 2
0023  2818 ;               e.  ROTATE THE LOOP COUNT LEFT BY 2 TO FORM A CACHE ADDRESS
0023  2819 ;               f.  LOAD THE PC WITH THE CACHE ADDRESS TO FORCE PREFETCH
0023  2820 ;               g.  PERFORM IRD1TEST TO LOAD THE IR REGISTER
0023  2821 ;               h.  SET THE PSL TO THE TEST CONDITIONS
0023  2822 ;               i.  BUT ON THE BRANCH CONDITIONS
0023  2823 ;       7.  TEST THE CS ADDRESS FOR THE CORRECT BRANCH
0023  2824 ;       8.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST PATTERNS
0023  2825 ;
0023  2826 ;
0023  2827 ; TEST PATTERNS:
0023  2828 ;
0023  2829 ;       ITERATION       IR      PSL
0023  2830 ;
0023  2831 ;           1           15      801F000F
0023  2832 ;           2           16      801F0000
0023  2833 ;           3           18      801F000A
0023  2834 ;           4           11      801F0002
0023  2835 ;           5           1A      801F0000
0023  2836 ;           6           17      801F0004
0023  2837 ;           7           10      801F000C
0023  2838 ;           8           1B      801F0001
0023  2839 ;           9           1C      801F0000
0023  2840 ;           A           10      801F0006
0023  2841 ;           B           1E      801F0000
0023  2842 ;           C           10      801F0008
0023  2843 ;           D           13      801F0009
0023  2844 ;           E           1E      801F0001
0023  2845 ;           F           10      801F000E
0023  2846 ;          10           11      801F000A
0023  2847 ;          11           19      801F0007
0023  2848 ;          12           1B      801F0004
0023  2849 ;          13           1D      801F0002
0023  2850 ;          14           16      801F0002
```

```
                        0023  2851 ;          15          1A      801F0004
                        0023  2852 ;          16          1B      801F0000
                        0023  2853 ;          17          1C      801F0002
                        0023  2854 ;          18          17      801F0000
                        0023  2855 ;          19          1A      801F0001
                        0023  2856 ;          1A          1D      801F0000
                        0023  2857 ;          1B          16      801F0004
                        0023  2858 ;          1C          17      801F0002
                        0023  2859 ;          1D          1F      801F0004
                        0023  2860 ;
                        0023  2861 ;
                        0023  2862 ; TEST ALGORITHM:
                        0023  2863 ;
                        0023  2864 ;       THESE TEST PATTERNS WERE GENERATED USING A LOGIC SIMULATOR.  SO IF
                        0023  2865 ;       YOU ATTEMPT TO READ THE VAX SRM AND TRY AN CORRELATE THE INSTRUCTIONS
                        0023  2866 ;       USED VERSUS BRANCH CONDITIONS IT REALLY WON'T MAKE ANY SENSE.  THE
                        0023  2867 ;       ATTEMPT HERE IS TO TEST THE INTERNAL WORKINGS OF THE CCC GATE ARRAY.
                        0023  2868 ;       ANY FAILUES SHOULD BE CONFINED TO THIS ARRAY OR THE INSTRUCTION
                        0023  2869 ;       REGISTER FEEDING IT.
                        0023  2870 ;
                        0023  2871 ; ASSUMPTIONS:
                        0023  2872 ;
                        0023  2873 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
                        0023  2874 ;
                        0023  2875 ; ERROR DESCRIPTION:
                        0023  2876 ;
                        0023  2877 ;       EXPECTED CS ADDRESS
                        0023  2878 ;       RECEIVED CS ADDRESS
                        0023  2879 ;       LOOP COUNT I
                        0023  2880 ;
                        0023  2881 ;--
                        0023  2882 ;*******************************************************************************
                        0023  2883
                        0023  2884
                        0023  2885          INITIALIZE                              ;INT THE CPU
                        0025  2886          LOOP            I,1,29                  ;CREATE A TEST LOOP OF 29
                        002C  2887          LDFIELD         4$,I,B,5$,34,39,        ;MODIFY U-CODE FOR THIS LOOP
                        003D  2888          LOADDCS         5$,,09,1                ;LOAD U-CODE INTO DCS
                        0044  2889          SAVEINDEX       I,3$                    ;SAVE LOOP COUNT
                        0049  2890          LOADREG         WDREG,3$,,L             ;PUT LOOP COUNT INTO WDREG
                        0051  2891          ERRLOOP                                 ;ERROR LOOP POINT
                        0053  2892          FETCH           0                       ;START DCS PROGRAM
                        0058  2893          BRSTCLK         <^X0B>                  ;END DCS PROGRAM
                        005C  2894          CMPREGMSK       CSTRP,1$,I,2$,,W,       ;TEST CS ADDRESS
                        0068  2895          IFERROR         ,<CCC>,                 ;ADDRESS INCORRECT
                        006E  2896          ENDLOOP         I                       ;END TEST LOOP
                        0071  2897          SKIP                                    ;GO TO NEXT TEST
                        0078  2898
                        0078  2899
                        0078  2900 BEGIN_TEST_DATA
                        0078
                        0078          ;-----------------------------------------------------------------------
                        0078  2901
1800                    0078  2902 1$:      .WORD   ^X1800                          ;RESULTS TABLE
1801                    007A  2903          .WORD   ^X1801
```

```
                     1800  007C  2904          .WORD   ^X1800
                     1801  007E  2905          .WORD   ^X1801
                     1801  0080  2906          .WORD   ^X1801
                     1801  0082  2907          .WORD   ^X1801
                     1800  0084  2908          .WORD   ^X1800
                     1801  0086  2909          .WORD   ^X1801
                     1801  0088  2910          .WORD   ^X1801
                     1801  008A  2911          .WORD   ^X1801                          ;LOOP 10
                     1801  008C  2912          .WORD   ^X1801
                     1801  008E  2913          .WORD   ^X1801
                     1800  0090  2914          .WORD   ^X1800
                     1800  0092  2915          .WORD   ^X1800
                     1801  0094  2916          .WORD   ^X1801
                     1801  0096  2917          .WORD   ^X1801
                     1800  0098  2918          .WORD   ^X1800
                     1801  009A  2919          .WORD   ^X1801
                     1801  009C  2920          .WORD   ^X1801
                     1800  009E  2921          .WORD   ^X1800                          ;LOOP 20
                     1800  00A0  2922          .WORD   ^X1800
                     1800  00A2  2923          .WORD   ^X1800
                     1800  00A4  2924          .WORD   ^X1800
                     1800  00A6  2925          .WORD   ^X1800
                     1800  00A8  2926          .WORD   ^X1800
                     1800  00AA  2927          .WORD   ^X1800
                     1800  00AC  2928          .WORD   ^X1800
                     1801  00AE  2929          .WORD   ^X1801
                     1800  00B0  2930          .WORD   ^X1800
                     3FFF  00B2  2931 2$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                 00000000  00B4  2932 3$:      .LONG   ^X00000000                      ;LOOP COUNT STORAGE
  07 00 04 06 02 00 03 05 00 0B  00B8  2933 4$:      .BYTE   11,00,05,03,00,02,06,04,00,07   ;RSRC LDFIELD TABLE
  03 03 02 0A 05 09 04 08 01 00  00C2  2934          .BYTE   00,01,08,04,09,05,10,02,03,03
     02 03 02 00 04 00 03 00 02  00CC  2935          .BYTE   02,00,03,00,04,00,02,03,02
                                 00D5  2936
                                 00D5  2937 5$:
U 09, 7700,C800,5BF4,032C,0070,180A  00D5  2938 ;% 09   PSL_R[TEMP11]                    ;U-CODE TO MODIFY
                                 00E0  2942
                                 00E0  2943
                                 00E0  2944
                                 00E0  2945 BEGIN_CPU_DATA
                                 0002  2946
                                 0002  2947 DCS_DATA
                                 0001  2948
U 00, 7700,C800,0364,0300,0470,1801  0001  2949 ;% 00   NOP
U 01, 7700,0800,5BF4,0330,0070,1802  000C  2953 ;% 01   PSL_R[TEMP12]                    ;IPL = 1F NATIVE MODE
U 02, 7700,8800,5BE4,0334,2470,1803  0017  2957 ;% 02   WCTRL/LOADCRAR,WB_R[TEMP13]      ;DISABLE ^P INTERRUPTS
U 03, 7700,8800,5BE4,0338,2C70,1804  0022  2961 ;% 03   WCTRL/CONWRITE,WB_R[MM.TEMP5]    ;...
U 04, 5700,0840,0364,033C,0470,1805  002D  2965 ;% 04   R[MM.TEMP1]_RDM                  ;LOOP COUNT TO RTEMPS
U 05, 7700,D800,EF64,0301,0470,1806  0038  2969 ;% 05   PL_[2]                          ;PLATCH GETS 2
U 06, 7700,8800,8B71,033C,0470,1807  0043  2973 ;% 06   Q_R[MM.TEMP1].RL.P              ;FORM CACHE ADDRESS
U 07, 7700,4800,03A4,03D8,4870,1808  004E  2977 ;% 07   WCTRL/PC_WB,WB_Q                 ;FORCE PREFETCH
U 08, 7700,4800,0364,1700,0470,1809  0059  2981 ;% 08   IRD1TEST                        ;LOAD INST. REGISTER
U 09, 7700,C800,5BF4,032C,0070,180A  0064  2985 ;% 09   PSL_R[TEMP11]                    ;SET COMP. MODE TEST CONDITIONS
U 0A, 7700,4800,0364,B701,8E70,1800  006F  2989 ;% 0A   BUT/CCBR,CCMISC/NOP.CCBR_BRATST,NEXT/1800;BUT ON BRANCH CONDITIONS
U 0B, 7B00,4800,0364,0300,0470,180C  007A  2993 ;% 0B   NOP,STOP.CLOCK,LATCH.CS.ADDR      ;STOP THE CPU CLOCK
                                 0085  2997
```

```
                    0085  2998 RTEMP_DATA
                    0001  2999
801F0000            0001  3000          .LONG   ^X801F0000                      ;TEST PSL'S
801F0008            0005  3001          .LONG   ^X801F0008
801F0004            0009  3002          .LONG   ^X801F0004
801F0002            000D  3003          .LONG   ^X801F0002
801F0001            0011  3004          .LONG   ^X801F0001
801F000A            0015  3005          .LONG   ^X801F000A
801F000C            0019  3006          .LONG   ^X801F000C
801F0006            001D  3007          .LONG   ^X801F0006
801F0009            0021  3008          .LONG   ^X801F0009
801F000E            0025  3009          .LONG   ^X801F000E
801F0007            0029  3010          .LONG   ^X801F0007
801F000F            002D  3011          .LONG   ^X801F000F
001F0000            0031  3012          .LONG   ^X001F0000
00C00000            0035  3013          .LONG   ^X00C00000                      ;DISABLE ^P INTERRUPTS
00400000            0039  3014          .LONG   ^X00400000                      ;...
00000000            003D  3015          .LONG   ^X00000000                      ;LOOP COUNT STORAGE
                    0041  3016
                    0041  3017 CACHE_DATA
                    0001  3018
00000015            0001  3019          .LONG   ^X00000015                      ;INSTRUCTION TABLE
00000016            0005  3020          .LONG   ^X00000016
00000018            0009  3021          .LONG   ^X00000018
00000011            000D  3022          .LONG   ^X00000011
0000001A            0011  3023          .LONG   ^X0000001A
00000017            0015  3024          .LONG   ^X00000017
00000010            0019  3025          .LONG   ^X00000010
0000001B            001D  3026          .LONG   ^X0000001B
0000001C            0021  3027          .LONG   ^X0000001C
00000010            0025  3028          .LONG   ^X00000010                      ;LOOP 10
0000001E            0029  3029          .LONG   ^X0000001E
00000010            002D  3030          .LONG   ^X00000010
00000013            0031  3031          .LONG   ^X00000013
0000001E            0035  3032          .LONG   ^X0000001E
00000010            0039  3033          .LONG   ^X00000010
00000011            003D  3034          .LONG   ^X00000011
00000019            0041  3035          .LONG   ^X00000019
0000001B            0045  3036          .LONG   ^X0000001B
00C0001D            0049  3037          .LONG   ^X0000001D
00C00016            004D  3038          .LONG   ^X00000016                      ;LOOP 20
0000001A            0051  3039          .LONG   ^X0000001A
0000001B            0055  3040          .LONG   ^X0000001B
0000001C            0059  3041          .LONG   ^X0000001C
00000017            005D  3042          .LONG   ^X00000017
0000001A            0061  3043          .LONG   ^X0000001A
0000001D            0065  3044          .LONG   ^X0000001D
00000016            0069  3045          .LONG   ^X00000016
00000017            006D  3046          .LONG   ^X00000017
0000001F            0071  3047          .LONG   ^X0000001F
00000000            0075  3048          .LONG   ^X00000000
                    0079  3049
                    0079  3050 END_CPU_DATA
                    00E0  3051
                    00E0  3052
```

```
                        00E0  3053 END_TEST_DATA
                        00E0
                        00E0       ;-----------------------------------------------------------------------------
                        00E0  3054 ENDTEST
```

```
                        002A        .SBTTL  "TEST   09B     TEST INTEGER OVERFLOW ARITHMETIC TRAP
                        002A  3056 ;*******************************************************************************
                        002A  3057 ;++
                        002A  3058 ;
                        002A  3059 ; FUNCTIONAL DESCRIPTION:
                        002A  3060 ;
                        002A  3061 ;       THIS TEST CHECKS THE INTEGER OVERFLOW LOGIC IN THE CCC GATE ARRAY.
                        002A  3062 ;
                        002A  3063 ;
                        002A  3064 ; TEST ALGORITHM:
                        002A  3065 ;
                        002A  3066 ;       1.  CREATE A TEST LOOP OF 20
                        002A  3067 ;       2.  SAVE THE LOOP COUNT
                        002A  3068 ;       3.  MOVE THE LOOP COUNT TO THE WDREG
                        002A  3069 ;       4.  EXECUTE THE DCS PROGRAM
                        002A  3070 ;               a.  DISABLE ^P INTERRUPTS
                        002A  3071 ;               b.  MOVE THE LOOP COUNT TO THE RTEMPS
                        002A  3072 ;               c.  PLATCH GETS 2
                        002A  3073 ;               d.  FORM CACHE ADDRESS FROM LOOP COUNT
                        002A  3074 ;               e.  LOAD PC TO CAUSE PREFETCH
                        002A  3075 ;               f.  LOAD INST. REGISTER WITH IRD1TEST
                        002A  3076 ;               g.  SET IV BIT IN PSL
                        002A  3077 ;               h.  SET V BIT IN PSL
                        002A  3078 ;               i.  PERFORM IRD1TEST TO ALLOW ARITHMETIC TRAP
                        002A  3079 ;       5.  TEST CS ADDRESS FOR CORRECT RESULTS
                        002A  3080 ;       6.  IF NO ERROR GO TO STEP 2 FOR REMAINING LOOP COUNTS
                        002A  3081 ;
                        002A  3082 ;
                        002A  3083 ; TEST PATTERNS:
                        002A  3084 ;
                        002A  3085 ;         ITERATION        IR
                        002A  3086 ;             1            38
                        002A  3087 ;             2            58
                        002A  3088 ;             3            F1
                        002A  3089 ;             4            49
                        002A  3090 ;             5            3C
                        002A  3091 ;             6            71
                        002A  3092 ;             7            F8
                        002A  3093 ;             8            5A
                        002A  3094 ;             9            50
                        002A  3095 ;             A            11
                        002A  3096 ;             B            40
                        002A  3097 ;             C            98
                        002A  3098 ;             D            A0
                        002A  3099 ;             E            08
                        002A  3100 ;             F            4C
                        002A  3101 ;            10            00
                        002A  3102 ;            11            2C
                        002A  3103 ;            12            52
                        002A  3104 ;            13            59
                        002A  3105 ;            14            78
                        002A  3106 ;
                        002A  3107 ;
                        002A  3108 ; LOGIC DESCRIPTION:
                        002A  3109 ;
```

```
                        002A  3110 ;       THIS TEST VERIFIES THAT THE INTERGER OVERFLOW LOGIC IN THE CCC GATE
                        002A  3111 ;       ARRAY IS OPERATIONAL.  ALL FAILURES SHOULD BE CONFINED TO THE ARRAY.
                        002A  3112 ;
                        002A  3113 ;
                        002A  3114 ; ASSUMPTIONS:
                        002A  3115 ;
                        002A  3116 ;       THIS TEST ASSUMES THE ARITHMETIC TRAP TEST HAS RUN SUCCESSFULLY
                        002A  3117 ;
                        002A  3118 ;
                        002A  3119 ; ERROR DESCRIPTION:
                        002A  3120 ;
                        002A  3121 ;       EXPECTED CS ADDRESS
                        002A  3122 ;       RECEIVED CD ADDRESS
                        002A  3123 ;       LOOP COUNT I
                        002A  3124 ;
                        002A  3125 ;--
                        002A  3126 ;*******************************************************************************
                        002A  3127
                        002A  3128
                        002A  3129          INITIALIZE                                  ;INIT THE CPU
                        002C  3130          EXPUTRAP                                    ;EXPECT U-TRAPS
                        002E  3131          LOOP            I,1,20                      ;TEST LOOP OF 20
                        0035  3132          SAVEINDEX       I,3$                        ;SAVE THE LOOP COUNT
                        003A  3133          LOADREG         WDREG,3$,,L                 ;MOVE LOOP COUNT TO WDREG
                        0042  3134          ERRLOOP                                     ;ERROR LOOP POINT
                        0044  3135          FETCH           0                           ;START DCS PROGRAM
                        0049  3136          BRSTCLK         <^X0C>                      ;END DCS PROGRAM
                        004D  3137          CMPREGMSK       CSTRP,1$,I,2$,,W,           ;TEST CS ADDRESS
                        0059  3138          IFERROR         ,<CCC>,                     ;ADDRESS INCORRECT
                        005F  3139          ENDLOOP         I                           ;END TEST LOOP
                        0062  3140          SKIP                                        ;GO TO NEXT TEST
                        0069  3141
                        0069  3142
                        0069  3143 BEGIN_TEST_DATA
                        0069
                        0069          ;-------------------------------------------------------------------------
                        0069  3144
180C                    0069  3145 1$:       .WORD   ^X180C                          ;RESULTS TABLE
0011                    006B  3146          .WORD   ^X0011
0011                    006D  3147          .WORD   ^X0011
0011                    006F  3148          .WORD   ^X0011
0011                    0071  3149          .WORD   ^X0011
180C                    0073  3150          .WORD   ^X180C
180C                    0075  3151          .WORD   ^X180C
180C                    0077  3152          .WORD   ^X180C
0011                    0079  3153          .WORD   ^X0011
0011                    007B  3154          .WORD   ^X0011                          ;LOOP 10
180C                    007D  3155          .WORD   ^X180C
0011                    007F  3156          .WORD   ^X0011
0011                    0081  3157          .WORD   ^X0011
180C                    0083  3158          .WORD   ^X180C
180C                    0085  3159          .WORD   ^X180C
180C                    0087  3160          .WORD   ^X180C
180C                    0089  3161          .WORD   ^X180C
180C                    008B  3162          .WORD   ^X180C
```

```
                          180C  008D  3163          .WORD   ^X180C
                          0011  008F  3164          .WORD   ^X0011                          ;LOOP 20
                          3FFF  0091  3165 2$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                      00000000  0093  3166 3$:      .LONG   ^X00000000                      ;LOOP COUNT STORAGE
                                0097  3167
                                0097  3168
                                0097  3169 BEGIN_CPU_DATA
                                0002  3170
                                0002  3171 DCS_DATA
                                0001  3172
U 00. 7700,C800,0364,0300,0470,1801   0001  3173 ;% 00    NOP
U 01. 7700,4800,5BE4,0300,2470,1802   000C  3177 ;% 01    WCTRL/LOADCRAR,WB_R[TEMP0]          ;KILL ^P INTERRUPTS
U 02. 7700,0800,5BE4,0304,2C70,1803   0017  3181 ;% 02    WCTRL/CONWRITE,WB_R[TEMP1]          ;...
U 03. 5700,C840,0364,0308,0470,1804   0022  3185 ;% 03    R[TEMP2]_RDM                        ;LOOP COUNT TO RTEMPS
U 04. 7700,D800,EF64,0301,0470,1805   002D  3189 ;% 04    PL_[2]                              ;PLATCH GETS 2
U 05. 7700,4800,8B71,0308,0470,1806   0038  3193 ;% 05    Q_R[TEMP2].RL.P                     ;CREATE CACHE ADDRESS
U 06. 7700,4800,03A4,03D8,4870,1807   0043  3197 ;% 06    WCTRL/PC_WB,WB_Q                    ;FORCE PREFETCH
U 07. 7700,C800,0364,1700,0470,1808   004E  3201 ;% 07    IRD1TEST                            ;LOAD IR REGISTER
U 08. 7700,8800,5BF4,030C,0070,1809   0059  3205 ;% 08    PSL_R[TEMP3]                        ;LOAD IV BIT IN PSL
U 09. 7700,0800,0364,0300,8E70,180A   0064  3209 ;% 09    SET V                               ;SET V BIT IN PSL
U 0A. 7700,C800,0364,1700,0470,180B   006F  3213 ;% 0A    IRD1TEST                            ;CHECK FOR ARITHMETIC TRAP
U 0B. 7700,4800,0364,0300,0470,180C   007A  3217 ;% 0B    NOP                                 ;...
U 0C. 7300,C800,0364,0300,0470,180D   0085  3221 ;% 0C    NOP,STOP.CLOCK                      ;STOP THE CPU CLOCK
                                0090  3225
                                0090  3226 RTEMP_DATA
                                0001  3227
                          00C00000  0001  3228          .LONG   ^X00C00000                  ;DISABLE ^P INTERRUPT
                          00400000  0005  3229          .LONG   ^X00400000                  ;...
                          00000000  0009  3230          .LONG   ^X00000000                  ;LOOP COUNT STORAGE
                          001F0020  000D  3231          .LONG   ^X001F0020                  ;IV BIT IN PSL
                                0011  3232
                                0011  3233 CACHE_DATA
                                0001  3234
                          00000038  0001  3235          .LONG   ^X00000038                  ;INSTRUCTIONS FOR CACHE
                          00000058  0005  3236          .LONG   ^X00000058
                          000000F1  0009  3237          .LONG   ^X000000F1
                          00000049  000D  3238          .LONG   ^X00000049
                          0000003C  0011  3239          .LONG   ^X0000003C
                          00000071  0015  3240          .LONG   ^X00000071
                          000000F8  0019  3241          .LONG   ^X000000F8
                          0000005A  001D  3242          .LONG   ^X0000005A
                          00000050  0021  3243          .LONG   ^X00000050
                          00000011  0025  3244          .LONG   ^X00000011                  ;LOOP 10
                          00000040  0029  3245          .LONG   ^X00000040
                          00000098  002D  3246          .LONG   ^X00000098
                          000000A0  0031  3247          .LONG   ^X000000A0
                          00000008  0035  3248          .LONG   ^X00000008
                          0000004C  0039  3249          .LONG   ^X0000004C
                          00000000  003D  3250          .LONG   ^X00000000
                          0000002C  0041  3251          .LONG   ^X0000002C
                          00000052  0045  3252          .LONG   ^X00000052
                          00000059  0049  3253          .LONG   ^X00000059
                          00000078  004D  3254          .LONG   ^X00000078                  ;LOOP 20
                          00000000  0051  3255          .LONG   ^X00000000
                                0055  3256
```

```
                        0055  3257 END_CPU_DATA
                        0097  3258
                        0097  3259
                        0097  3260 END_TEST_DATA
                        0097
                        0097       ;-----------------------------------------------------------------------
                        0097  3261 ENDTEST
```

```
                        002A        .SBTTL  "TEST   09C     TEST DECIMAL OVERFLOW ARITHMETIC TRAP
                        002A  3263 ;*******************************************************************************
                        002A  3264 ;++
                        002A  3265 ;
                        002A  3266 ; FUNCTIONAL DESCRIPTION:
                        002A  3267 ;
                        002A  3268 ;       THIS TEST CHECKS THE DECIMAL OVERFLOW LOGIC IN THE CCC GATE ARRAY.
                        002A  3269 ;
                        002A  3270 ;
                        002A  3271 ; TEST ALGORITHM:
                        002A  3272 ;
                        002A  3273 ;       1.  CREATE A TEST LOOP OF 13
                        002A  3274 ;       2.  SAVE THE LOOP COUNT
                        002A  3275 ;       3.  MOVE THE LOOP COUNT TO THE WDREG
                        002A  3276 ;       4.  EXECUTE DCS PROGRAM
                        002A  3277 ;               a.  DISABLE ^P INTERRUPTS
                        002A  3278 ;               b.  MOVE LOOP COUNT TO RTEMPS
                        002A  3279 ;               c.  FORM CACHE ADDRESS
                        002A  3280 ;               d.  LOAD PC TO CAUSE PREFETCH
                        002A  3281 ;               e.  PERFORM IRD1TEST TO LOAD INSTRUCTION REGISTER
                        002A  3282 ;               f.  SET DV BIT IN PSL
                        002A  3283 ;               g.  SET V BIT IN PSL
                        002A  3284 ;               h.  PERFORM IRD1TEST TO ALLOW ARITHMETIC TRAP
                        002A  3285 ;       5.  TEST CS ADDRESS FOR CORRECT RESULTS
                        002A  3286 ;       6.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING LOOP COUNTS
                        002A  3287 ;       7.  EXECUTE DCS PROGRAM
                        002A  3288 ;               a.  READ ATCR REGISTER TO RDM
                        002A  3289 ;       8.  TEST ATCR REGISTER FOR DV ARITHMETIC TRAP
                        002A  3290 ;       9.  IF NO ERROR GO TO NEXT TEST
                        002A  3291 ;
                        002A  3292 ;
                        002A  3293 ; TEST PATTERNS:
                        002A  3294 ;
                        002A  3295 ;       ITERATION       IR
                        002A  3296 ;           1           20
                        002A  3297 ;           2           68
                        002A  3298 ;           3           E8
                        002A  3299 ;           4           2C
                        002A  3300 ;           5           08
                        002A  3301 ;           6           30
                        002A  3302 ;           7           A8
                        002A  3303 ;           8           00
                        002A  3304 ;           9           60
                        002A  3305 ;           A           A0
                        002A  3306 ;           B           C8
                        002A  3307 ;           C           E0
                        002A  3308 ;           D           EC
                        002A  3309 ;
                        002A  3310 ;
                        002A  3311 ; LOGIC DESCRIPTION:
                        002A  3312 ;
                        002A  3313 ;       THIS TEST EXERCISES THE DECIMAL OVERFLOW LOGIC IN THE CCC GATE ARRAY.
                        002A  3314 ;       ALL ERRORS SHOULD BE CONFINED TO THIS GATE ARRAY.
                        002A  3315 ;
                        002A  3316 ;
```

```
                        002A  3317 ; ASSUMPTIONS:
                        002A  3318 ;
                        002A  3319 ;       THIS TEST ASSUMES THE ARITHMETIC TRAP TEST HAS RUN SUCCESSFULLY
                        002A  3320 ;
                        002A  3321 ;
                        002A  3322 ; ERROR DESCRIPTION:
                        002A  3323 ;
                        002A  3324 ;       FIRST IFERROR:
                        002A  3325 ;               EXPECTED CS ADDRESS
                        002A  3326 ;               RECEIVED CS ADDRESS
                        002A  3327 ;               LOOP COUNT I
                        002A  3328 ;
                        002A  3329 ;       SECOND IFERROR:
                        002A  3330 ;               EXPECTED ATCR REGISTER CONTENTS
                        002A  3331 ;               RECEIVED ATCR REGISTER CONTENTS
                        002A  3332 ;
                        002A  3333 ;--
                        002A  3334 ;*******************************************************************************
                        002A  3335
                        002A  3336
                        002A  3337          INITIALIZE                              ;INIT THE CPU
                        002C  3338          EXPUTRAP                                ;EXPECT U-TRAPS
                        002E  3339          LOOP            I,1,13                  ;TEST LOOP OF 13
                        0035  3340          SAVEINDEX       I,3$                    ;SAVE THE LOOP COUNT
                        003A  3341          LOADREG         WDREG,3$,,L             ;MOVE LOOP COUNT TO WDREG
                        0042  3342          ERRLOOP                                 ;ERROR LOOP POINT
                        0044  3343          FETCH           0                       ;START DCS PROGRAM
                        0049  3344          BRSTCLK         <^X0C>                  ;END DCS PROGRAM
                        004D  3345          CMPREGMSK       CSTRP,1$,I,2$,,W,       ;TEST CS ADDRESS
                        0059  3346          IFERROR         ,<CCC>,                 ;ADDRESS INCORRECT
                        005F  3347          ENDLOOP         I                       ;END TEST LOOP
                        0062  3348          FETCH           <^X0D>                  ;START DCS PROGRAM
                        0067  3349          BRSTCLK         <^X0F>                  ;END DCS PROGRAM
                        006B  3350          CMPREGMSK       WHREG,4$,,5$,,L,        ;TEST ATCP CONTENTS
                        0077  3351          IFERROR         ,<CCC>,                 ;REGISTER INCORRECT
                        007D  3352          SKIP                                    ;GO TO NEXT TEST
                        0084  3353
                        0084  3354
                        0084  3355 BEGIN_TEST_DATA
                        0084
                        0084          ;-----------------------------------------------------------------------
                        0084  3356
0011                    0084  3357 1$:       .WORD   ^X0011                      ;RESULTS TABLE
180C                    0086  3358           .WORD   ^X180C
0011                    0088  3359           .WORD   ^X0011
180C                    008A  3360           .WORD   ^X180C
0011                    008C  3361           .WORD   ^X0011
180C                    008E  3362           .WORD   ^X180C
180C                    0090  3363           .WORD   ^X180C
180C                    0092  3364           .WORD   ^X180C
180C                    0094  3365           .WORD   ^X180C
180C                    0096  3366           .WORD   ^X180C                      ;LOOP 10
180C                    0098  3367           .WORD   ^X180C
180C                    009A  3368           .WORD   ^X180C
180C                    009C  3369           .WORD   ^X180C
```

```
                      3FFF  009E  3370 2$:     .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                  00000000  00A0  3371 3$:     .LONG   ^X00000000                      ;LOOP COUNT STORAGE
                  00000006  00A4  3372 4$:     .LONG   ^X00000006                      ;ATCR CONTENTS
                  00000007  00A8  3373 5$:     .LONG   ^X00000007                      ;MASK FOR CMPREGMSK PSEUDO OP
                            00AC  3374
                            00AC  3375
                            00AC  3376 BEGIN_CPU_DATA
                            0002  3377
                            0002  3378 DCS_DATA
                            0001  3379
U 00, 7700,C800,0364,0300,0470,1801 0001  3380 ;* 00   NOP
U 01, 7700,4800,5BE4,0300,2470,1802 000C  3384 ;* 01   WCTRL/LOADCRAR,WB_R[TEMP0]          ;KILL ^P INTERRUPTS
U 02, 7700,0800,5BE4,0304,2C70,1803 0017  3388 ;* 02   WCTRL/CONWRITE,WB_R[TEMP1]         ;...
U 03, 5700,C840,0364,0308,0470,1804 0022  3392 ;* 03   R[TEMP2]_RDM                       ;LOOP COUNT TO RTEMPS
U 04, 7700,D800,EF64,0301,0470,1805 002D  3396 ;* 04   PL_[2]                            ;PLATCH GETS 2
U 05, 7700,4800,8B71,0308,0470,1806 0038  3400 ;* 05   Q_R[TEMP2].RL.P                   ;CREATE CACHE ADDRESS
U 06, 7700,4800,03A4,03D8,4870,1807 0043  3404 ;* 06   WCTRL/PC_WB,WB_Q                   ;FORCE PREFETCH
U 07, 7700,C800,0364,1700,0470,1808 004E  3408 ;* 07   IRD1TEST                          ;LOAD IR REGISTER
U 08, 7700,8800,5BF4,030C,0070,1809 0059  3412 ;* 08   PSL_R[TEMP3]                      ;LOAD DV BIT IN PSL
U 09, 7700,0800,0364,0300,8E70,180A 0064  3416 ;* 09   SET V                            ;SET V BIT IN PSL
U 0A, 7700,C800,0364,1700,0470,180B 006F  3420 ;* 0A   IRD1TEST                          ;CHECK FOR ARITHMETIC TRAP
U 0B, 7700,4800,0364,0300,0470,180C 007A  3424 ;* 0B   NOP                              ;...
U 0C, 7300,C800,0364,0300,0470,180D 0085  3428 ;* 0C   NOP,STOP.CLOCK                   ;STOP THE CPU CLOCK
U 0D, 7700,C800,0364,0300,0470,180E 0090  3432 ;* 0D   NOP
U 0E, 6700,4800,0364,0300,0E70,180F 009B  3436 ;* 0E   RDM_WB,CCMISC/WB_ATCR.CCBR_SIGND  ;READ ATCR REGISTER TO RDM
U 0F, 7300,C800,0364,0300,0470,1810 00A6  3440 ;* 0F   NOP,STOP.CLOCK                   ;STOP THE CPU CLOCK
                            00B1  3444
                            00B1  3445 RTEMP_DATA
                            0001  3446
                  00C00000  0001  3447         .LONG   ^X00C00000                      ;DISABLE ^P INTERRUPT
                  00400000  0005  3448         .LONG   ^X00400000                      ;...
                  00000000  0009  3449         .LONG   ^X00000000                      ;LOOP COUNT STORAGE
                  001F0080  000D  3450         .LONG   ^X001F0080                      ;DV BIT IN PSL
                            0011  3451
                            0011  3452 CACHE_DATA
                            0001  3453
                  00000020  0001  3454         .LONG   ^X00000020                      ;INSTRUCTIONS FOR CACHE
                  00000068  0005  3455         .LONG   ^X00000068
                  000000E8  0009  3456         .LONG   ^X000000E8
                  0000002C  000D  3457         .LONG   ^X0000002C
                  00000008  0011  3458         .LONG   ^X00000008
                  00000030  0015  3459         .LONG   ^X00000030
                  000000A8  0019  3460         .LONG   ^X000000A8
                  00000000  001D  3461         .LONG   ^X00000000
                  00000060  0021  3462         .LONG   ^X00000060
                  000000A0  0025  3463         .LONG   ^X000000A0                      ;LOOP 10
                  000000C8  0029  3464         .LONG   ^X000000C8
                  000000E0  002D  3465         .LONG   ^X000000E0
                  000000EC  0031  3466         .LONG   ^X000000EC
                  00000000  0035  3467         .LONG   ^X00000000
                            0039  3468
                            0039  3469 END_CPU_DATA
                            00AC  3470
                            00AC  3471
                            00AC  3472 END_TEST_DATA
```

```
00AC
00AC        ;-----------------------------------------------------------------------
00AC  3473 ENDTEST
```

```
                        001B        .SBTTL  "TEST   09D     ALU STATE FLAGS TEST 1
                        001B  3475 ;*******************************************************************************
                        001B  3476 ;++
                        001B  3477 ;
                        001B  3478 ; FUNCTIONAL DESCRIPTION:
                        001B  3479 ;
                        001B  3480 ;       THIS TEST CHECKS THE ALU STATE LATCHES IN THE CCC GATE ARRAY.
                        001B  3481 ;       THE MICRO ORDERS USED ARE:
                        001B  3482 ;               1.  CCMISC/ALUS_SIGND.CCBR_ALUS
                        001B  3483 ;               2.  CCMISC/ALUS_UNSGN.CCBR_ALUS
                        001B  3484 ;               3.  CCMISC/ALUS_DSDZ.CCBR_ALUS
                        001B  3485 ;
                        001B  3486 ;
                        001B  3487 ; TEST ALGORITHM:
                        001B  3488 ;
                        001B  3489 ;       1.  CREATE A TEST LOOP FOR THE SUBTEST TEST
                        001B  3490 ;       2.  MODIFY U-CODE FOR THIS LOOP
                        001B  3491 ;       3.  LOAD NEW U-CODE INTO RDM DCS
                        001B  3492 ;       4.  EXECUTE DCS PROGRAM
                        001B  3493 ;               a.  Q REGISTER GETS TEST PATTERN
                        001B  3494 ;               b.  PERFORM ADD OF Q AND ONE RTEMP REGISTER, SET ALUS FLAGS
                        001B  3495 ;               c.  BRANCH CS ADDRESS ACCORDING TO ALUS FLAGS
                        001B  3496 ;       5.  TEST CS ADDRESS TO SEE IF FLAGS WHERE SET CORRECTLY
                        001B  3497 ;       6.  IF NO ERROR GO TO STEP 2 FOR REMAINING LOOP COUNTS
                        001B  3498 ;
                        001B  3499 ;
                        001B  3500 ; TEST PATTERNS:
                        001B  3501 ;
                        001B  3502 ;                               ITERATION       Q REG. + R[TEMP*]       CS ADDRESS
                        001B  3503 ;       SUBTEST #1:
                        001B  3504 ;                                   1           00000080 + 00000000         1802
                        001B  3505 ;                                   2           00000000 + 00000000         1801
                        001B  3506 ;
                        001B  3507 ;       SUBTEST #2
                        001B  3508 ;                                   1           00000000 + 00000000         1803
                        001B  3509 ;                                   2           000000C0 + 000000C0         1800
                        001B  3510 ;
                        001B  3511 ;       SUBTEST #3
                        001B  3512 ;                                   1           00000000 + 00000000         1801
                        001B  3513 ;                                   2           40404001 + 00000000         1802
                        001B  3514 ;                                   3           00000013 + 00000000         1802
                        001B  3515 ;                                   4           00000025 + 00000000         1802
                        001B  3516 ;                                   5           00000040 + 00000000         1800
                        001B  3517 ;                                   6           00000080 + 00000000         1800
                        001B  3518 ;
                        001B  3519 ;
                        001B  3520 ; LOGIC DESCRIPTION:
                        001B  3521 ;
                        001B  3522 ;       THIS TEST CHECKS THE INTERNAL LOGIC OF THE CCC GATE ARRAY THAT IS
                        001B  3523 ;       RESPONSIBLE FOR SETTING THE ALU STATE LATCHES.  ANY FAILURES SHOULD
                        001B  3524 ;       BE CONFINED TO THE ARRAY OR THE CCC ROM THAT SUPPLIES THE CCC CONTROL
                        001B  3525 ;       FIELD.
                        001B  3526 ;
                        001B  3527 ;
                        001B  3528 ; ASSUMPTIONS:
```

```
                        001B  3529 ;
                        001B  3530 ;       THIS TEST ASSUMES CONDITION CODE TESTS (N,Z,V,C) HAVE BEEN RUN.
                        001B  3531 ;
                        001B  3532 ;
                        001B  3533 ; ERROR DESCRIPTION:
                        001B  3534 ;
                        001B  3535 ;       EXPECTED CS ADDRESS
                        001B  3536 ;       RECEIVED CS ADDRESS
                        001B  3537 ;       LOOP COUNT I
                        001B  3538 ;
                        001B  3539 ;--
                        001B  3540 ;*******************************************************************************
                        001B  3541 ;///////////////////////////////////////////////////////////////////////////////
                        001B  3542 ;+
                        001B  3543 ;SUBTEST #1
                        001B  3544 ;
                        001B  3545 ;       CCMISC/ALUS_SIGND.CCBR_ALUS
                        001B  3546 ;-
                        001B  3547         SUBTEST                                         ;SUBTEST #1
                        001E
                        001E       ;////////////////////////////////////////////////////////////////////////////
                        001E  3548         INITIALIZE                                      ;INIT THE CPU
                        0020  3549         LOOP            I,1,2                           ;TEST LOOP OF 2
                        0027  3550         LDFIELD         1$,I,B,3$,34,39                 ;MODIFY U-CODE FOR THIS LOOP
                        0038  3551         LOADDCS         3$,,01,2                        ;LOAD U-CODE INTO DCS
                        003F  3552         ERRLOOP                                         ;ERROR LOOP POINT
                        0041  3553         FETCH           0                               ;START DCS PROGRAM
                        0046  3554         BRSTCLK         4                               ;END DCS PROGRAM
                        004A  3555         CMPREGMSK       CSTRP,5$,I,6$,,W,               ;TEST CS ADDRESS
                        0056  3556         IFERROR         ,<CCC>,                         ;ADDRESS INCORRECT
                        005C  3557         ENDLOOP         I                               ;END TEST LOOP
                        005F  3558
                        005F  3559
                        005F  3560
                        005F  3561 ;///////////////////////////////////////////////////////////////////////////////
                        005F  3562 ;+
                        005F  3563 ;SUBTEST #2
                        005F  3564 ;
                        005F  3565 ;       CCMISC/ALUS_UNSGN.CCBR_ALUS
                        005F  3566 ;-
                        005F  3567         SUBTEST                                         ;SUBTEST #2
                        0062
                        0062       ;////////////////////////////////////////////////////////////////////////////
                        0062  3568         INITIALIZE                                      ;INIT THE CPU
                        0064  3569         LOOP            I,1,2                           ;TEST LOOP OF 2
                        006B  3570         LDFIELD         7$,I,B,9$,34,39                 ;MODIFY U-CODE FOR THIS LOOP
                        007C  3571         LDFIELD         8$,I,B,10$,34,39                ;...
                        008D  3572         LOADDCS         9$,,01,2                        ;LOAD U-CODE INTO DCS
                        0094  3573         ERRLOOP                                         ;ERROR LOOP POINT
                        0096  3574         FETCH           0                               ;START DCS PROGRAM
                        009B  3575         BRSTCLK         4                               ;END DCS PROGRAM
                        009F  3576         CMPREGMSK       CSTRP,11$,I,6$,,W,              ;TEST CS ADDRESS
                        00AB  3577         IFERROR         ,<CCC>,                         ;ADDRESS INCORRECT
                        00B1  3578         ENDLOOP         I                               ;END TEST LOOP
                        00B4  3579
```

```
                        00B4  3580
                        00B4  3581
                        00B4  3582 ;////////////////////////////////////////////////////////////////////////////////
                        00B4  3583 ;+
                        00B4  3584 ;SUBTEST #3
                        00B4  3585 ;
                        00B4  3586 ;       CCMISC/ALUS_DSDZ.CCBR_ALUS
                        00B4  3587 ;-
                        00B4  3588         SUBTEST                                         ;SUBTEST #3
                        00B7
                        00B7       ;////////////////////////////////////////////////////////////////////////////////
                        00B7  3589         INITIALIZE                                      ;INIT THE CPU
                        00B9  3590         LOOP            I,1,6                           ;TEST LOOP OF 6
                        00C0  3591         LDFIELD         12$,I,B,14$,34,39               ;MODIFY U-CODE FOR THIS LOOP
                        00D1  3592         LOADDCS         14$,,01,2                       ;LOAD U-CODE INTO DCS
                        00D8  3593         ERRLOOP                                         ;ERROR LOOP POINT
                        00DA  3594         FETCH           0                               ;START DCS PROGRAM
                        00DF  3595         BRSTCLK         4                               ;END DCS PROGRAM
                        00E3  3596         CMPREGMSK       CSTRP,15$,I,6$,,W,              ;TEST CS ADDRESS
                        00EF  3597         IFERROR         ,<CCC>,                         ;ADDRESS INCORRECT
                        00F5  3598         ENDLOOP         I                               ;END TEST LOOP
                        00F8  3599         SKIP                                            ;GO TO NEXT TEST
                        00FF  3600
                        00FF  3601
                        00FF  3602 BEGIN_TEST_DATA
                        00FF
                        00FF       ;-------------------------------------------------------------------------
                        00FF  3603
                     00 01  00FF  3604 1$:      .BYTE   01,00                           ;RSRC FIELD SUBTEST #1
                        0101  3605 3$:
U 01, 7700,C800,5BE5,0304,0470,1802  0101  3606 ;% 01   Q_R[TEMP1]                       ;U-CODE SUBTEST #1
U 02, 7701,8800,0390,0001,8C70,1803  010C  3610 ;% 02   SIZE[BYTE],ALU/A+B+CI,MUX/R.Q,RSRC/TEMP0,CCMISC/ALUS_SIGND.CCBR_ALUS
                           1802  0117  3614 5$:      .WORD   ^X1802                          ;RESULTS SUBTEST #1
                           1801  0119  3615          .WORD   ^X1801                          ;...
                           3FFF  011B  3616 6$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                        011D  3617
                        02 00  011D  3618 7$:      .BYTE   00,02                           ;RSRC FIELD SUBTEST #2
                        02 00  011F  3619 8$:      .BYTE   00,02                           ;...
                        0121  3620 9$:
U 01, 7700,8800,5BE5,0300,0470,1802  0121  3621 ;% 01   Q_R[TEMP0]                       ;U-CODE SUBTEST #2
                        012C  3625 10$:
U 02, 7701,C800,0390,0001,0C70,1803  012C  3626 ;% 02   SIZE[BYTE],ALU/A+B+CI,MUX/R.Q,RSRC/TEMP0,CCMISC/ALUS_UNSGN.CCBR_ALUS
                           1803  0137  3630 11$:     .WORD   ^X1803                          ;RESULTS SUBTEST #2
                           1800  0139  3631          .WORD   ^X1800                          ;...
                        013B  3632
                  01 06 05 04 03 00  013B  3633 12$:     .BYTE   00,03,04,05,06,01               ;RSRC FIELD SUBTEST #3
                        0141  3634 14$:
U 01, 7700,8800,5BE5,0300,0470,1802  0141  3635 ;% 01   Q_R[TEMP0]                       ;U-CODE SUBTEST #3
U 02, 7701,8800,0390,0000,0C70,1803  014C  3639 ;% 02   SIZE[BYTE],ALU/A+B+CI,MUX/R.Q,RSRC/TEMP0,CCMISC/ALUS_DSDZ.CCBR_ALUS
                           1801  0157  3643 15$:     .WORD   ^X1801                          ;RESULTS SUBTEST #3
                           1802  0159  3644          .WORD   ^X1802
                           1802  015B  3645          .WORD   ^X1802
                           1802  015D  3646          .WORD   ^X1802
                           1800  015F  3647          .WORD   ^X1800
                           1800  0161  3648          .WORD   ^X1800
```

```
                                0163  3649
                                0163  3650
                                0163  3651 BEGIN_CPU_DATA
                                0002  3652
                                0002  3653 DCS_DATA
                                0001  3654
U 00, 7700,C800,0364,0300,0470,1801  0001  3655 ;% 00   NOP
U 01, 7700,8800,5BE5,0300,0470,1802  000C  3659 ;% 01   Q_R[TEMP0]                                        ;TEST DATA TO Q REGISTER
U 02, 7701,8800,0390,0001,8C70,1803  0017  3663 ;% 02   SIZE[BYTE],ALU/A+B+CI,MUX/R.Q,RSRC/TEMP0,CCMISC/ALUS_SIGND.CCBR_ALUS
                                0022  3667                                                             ;SET ALU FLAGS ACCORDING TO ADD
U 03, 7700,4800,0364,B701,8470,1800  0022  3668 ;% 03   ALUS?,NEXT/1800                                 ;BRANCH ON FLAGS
U 04, 7300,4800,0364,0300,0470,1805  002D  3672 ;% 04   NOP,STOP.CLOCK                                   ;STOP THE CPU CLOCK
                                0038  3676
                                0038  3677 RTEMP_DATA
                                0001  3678
                    00000000    0001  3679          .LONG   ^X00000000                                  ;TEST DATA
                    00000080    0005  3680          .LONG   ^X00000080
                    000000C0    0009  3681          .LONG   ^X000000C0
                    40404001    000D  3682          .LONG   ^X40404001
                    00000013    0011  3683          .LONG   ^X00000013
                    00000025    0015  3684          .LONG   ^X00000025
                    00000040    0019  3685          .LONG   ^X00000040
                                001D  3686
                                001D  3687 END_CPU_DATA
                                0163  3688
                                0163  3689
                                0163  3690 END_TEST_DATA
                                0163
                                0163        ;-------------------------------------------------------------------------------
                                0163  3691 ENDTEST
```

```
001B       .SBTTL  "TEST   09E     ALU STATE FLAGS TEST 2
001B  3693 ;*****************************************************************************
001B  3694 ;++
001B  3695 ;
001B  3696 ; FUNCTIONAL DESCRIPTION:
001B  3697 ;
001B  3698 ;       THIS IS A SECOND TEST OF THE ALU STATE LATCHES IN THE CCC GATE ARRAY.
001B  3699 ;       THE MICRO ORDERS USED HERE ARE:
001B  3700 ;               1.  CC/NOP.CCBR_SIGND
001B  3701 ;               2.  CCMISC/NOP.CCBR_CSIGNS
001B  3702 ;
001B  3703 ;
001B  3704 ; TEST ALGORITHM:
001B  3705 ;
001B  3706 ;       1.  CREATE A TEST LOOP OF 2
001B  3707 ;       2.  MODIFY U-CODE FOR THIS LOOP
001B  3708 ;       3.  LOAD NEW U-CODE INTO DCS
001B  3709 ;       4.  EXECUTE DCS PROGRAM
001B  3710 ;
001B  3711 ;               SUBTEST #1:
001B  3712 ;               a.  PUT TEST DATA IN Q REGISTER
001B  3713 ;               b.  PERFORM ADD OF Q AND AN RTEMP REG. WHILE BRANCHING ON CCBR
001B  3714 ;
001B  3715 ;               SUBTEST #2:
001B  3716 ;               a.  SOURCE TEST DATA TO WBUS AND BRANCH ON CCBR
001B  3717 ;
001B  3718 ;       5.  TEST CS ADDRESS FOR CORRECT BRANCH
001B  3719 ;       6.  IF NO ERROR GO TO STEP 2 FOR REMAINING LOOP COUNTS
001B  3720 ;
001B  3721 ;
001B  3722 ; TEST PATTERNS:
001B  3723 ;
001B  3724 ;       SUBTEST #1:
001B  3725 ;                       ITERATION       Q REG + R[TEMP1]        CS ADDRESS
001B  3726 ;                           1           00000080 + 00000000         1802
001B  3727 ;                           2           00000000 + 00000000         1801
001B  3728 ;
001B  3729 ;       SUBTEST #2:
001B  3730 ;                       ITERATION         WBUS              CS ADDRESS
001B  3731 ;                           1           00000080              1801
001B  3732 ;                           2           00000000              1800
001B  3733 ;
001B  3734 ;
001B  3735 ; LOGIC DESCRIPTION:
001B  3736 ;
001B  3737 ;       THIS TEST CHECKS THAT THE ALU STATE LATCHES IN THE CCC GATE ARRAY
001B  3738 ;       ARE OPERATIONAL.  ANY FAILURES SHOULD BE CONFINED TO THE CCC GATE
001B  3739 ;       ARRAY OR THE CCC ROM THAT SUPPLIES THE CCC CONTROL SIGNALS.
001B  3740 ;
001B  3741 ;
001B  3742 ; ASSUMPTIONS:
001B  3743 ;
001B  3744 ;       THIS TEST ASSUMES THAT THE CONDITION CODE TESTS (N,Z,V,C) HAVE RUN
001B  3745 ;
001B  3746 ;
```

```
                        ; ERROR DESCRIPTION:
                        ;
                        ;       EXPECTED CS ADDRESS
                        ;       RECEIVED CS ADDRESS
                        ;       LOOP COUNT I
                        ;
                        ;--
                        ;**************************************************************************************
                        ;//////////////////////////////////////////////////////////////////////////////////////
                        ;+
                        ;SUBTEST #1
                        ;
                        ;       CC/NOP.CCBR_SIGND
                        ;-
001B  3761              SUBTEST                                         ;SUBTEST #1
001E
001E                    ;//////////////////////////////////////////////////////////////////////////////////////
001E  3762              INITIALIZE                                      ;INIT THE CPU
0020  3763              LOOP            I,1,2                           ;TEST LOOP OF 2
0027  3764              LDFIELD         1$,I,B,4$,34,39                 ;MODIFY U-CODE FOR THIS LOOP
0038  3765              LOADDCS         3$,,00,4                        ;LOAD U-CODE INTO DCS
003F  3766              ERRLOOP                                         ;ERROR LOOP POINT
0041  3767              FETCH           0                               ;START DCS PROGRAM
0046  3768              BRSTCLK         3                               ;END DCS PROGRAM
004A  3769              CMPREGMSK       CSTRP,6$,I,7$,,W,               ;TEST CS ADDRESS
0056  3770              IFERROR         ,<CCC>,                         ;ADDRESS INCORRECT
005C  3771              ENDLOOP         I                               ;END TEST LOOP
005F  3772
005F  3773
005F  3774              ;//////////////////////////////////////////////////////////////////////////////////////
005F  3775              ;+
005F  3776              ;SUBTEST #2
005F  3777              ;
005F  3778              ;       CCMISC/NOP.CCBR_CSIGNS
005F  3779              ;-
005F  3780              SUBTEST                                         ;SUBTEST #2
0062
0062                    ;//////////////////////////////////////////////////////////////////////////////////////
0062  3781              LOOP            I,1,2                           ;TEST LOOP OF 2
0069  3782              LDFIELD         8$,I,B,10$,34,39                ;MODIFY U-CODE FOR THIS LOOP
007A  3783              LOADDCS         9$,,00,3                        ;LOAD U-CODE INTO DCS
0081  3784              ERRLOOP                                         ;ERROR LOOP POINT
0083  3785              FETCH           0                               ;START DCS PROGRAM
0088  3786              BRSTCLK         2                               ;END DCS PROGRAM
008C  3787              CMPREGMSK       CSTRP,11$,I,12$,,W,             ;TEST CS ADDRESS
0098  3788              IFERROR         ,<CCC>,                         ;ADDRESS INCORRECT
009E  3789              ENDLOOP         I                               ;END TEST LOOP
00A1  3790              SKIP                                            ;GO TO NEXT TEST
00A8  3791
00A8  3792
00A8  3793
00A8  3794 BEGIN_TEST_DATA
00A8
00A8                    ;-------------------------------------------------------------------------
00A8  3795
```

```
                          01 00  00A8  3796 1$:     .BYTE   00,01                          ;RSRC FIELD FOR SUBTEST #1
                                 00AA  3797 3$:
U 00, 7700,C800,0364,0300,0470,1801  00AA  3798 ;% 00   NOP                                    ;U-CODE SUBTEST #1
                                 00B5  3802 4$:
U 01, 7700,8800,5BE5,0300,0470,1802  00B5  3803 ;% 01   Q_R[TEMP0]                             ;MOVE TEST DATA TO Q REGISTER
U 02, 7701,4800,0390,B404,0470,1800  00C0  3807 ;% 02   SIZE[BYTE],ALU/A+B+CI,MUX/R.Q,RSRC/TEMP1,SIGND CMP?,NEXT/1800
                                 00CB  3811                                                ;SIGNED COMPARE
U 03, 7300,C800,0364,0300,0470,1804  00CB  3812 ;% 03   NOP,STOP.CLOCK                         ;STOP THE CPU CLOCK
                            1802  00D6  3816 6$:     .WORD   ^X1802                         ;RESULTS SUBTEST #1
                            1801  00D8  3817         .WORD   ^X1801
                            3FFF  00DA  3818 7$:     .WORD   ^X3FFF                         ;MASK FOR CMPREGMSK PSEUDO OP
                                 00DC  3819
                          01 00  00DC  3820 8$:     .BYTE   00,01                          ;RSRC FIELD FOR SUBTEST #2
                                 00DE  3821 9$:
U 00, 7700,C800,0364,0300,0470,1801  00DE  3822 ;% 00   NOP                                    ;U-CODE FOR SUBTEST #2
                                 00E9  3826 10$:
U 01, 7701,8800,5BE4,B400,8C70,1800  00E9  3827 ;% 01   SIZE[BYTE],WB_R[TEMP0],CMP SIGNS?,NEXT/1800
                                 00F4  3831                                                ;COMPARE SIGNS
U 02, 7300,4800,0364,0300,0470,1803  00F4  3832 ;% 02   NOP,STOP.CLOCK                         ;STOP THE CPU CLOCK
                            1801  00FF  3836 11$:    .WORD   ^X1801                         ;RESULTS SUBTEST #2
                            1800  0101  3837         .WORD   ^X1800
                            3FFD  0103  3838 12$:    .WORD   ^X3FFD                         ;MASK FOR CMPREGMSK PSEUDO OP
                                 0105  3839
                                 0105  3840
                                 0105  3841 BEGIN_CPU_DATA
                                 0002  3842
                                 0002  3843 RTEMP_DATA
                                 0001  3844
                        00000080  0001  3845         .LONG   ^X00000080                     ;TEST DATA
                        00000000  0005  3846         .LONG   ^X00000000
                                 0009  3847
                                 0009  3848 END_CPU_DATA
                                 0105  3849
                                 0105  3850
                                 0105  3851 END_TEST_DATA
                                 0105
                                 0105       ;-----------------------------------------------------------------------
                                 0105  3852 ENDTEST
```

```
                        001D        .SBTTL  "TEST   09F     TEST BUT/CCBR1.CCBR0.IR0
                        001D  3854 ;*************************************************************************
                        001D  3855 ;++
                        001D  3856 ;
                        001D  3857 ; FUNCTIONAL DESCRIPTION:
                        001D  3858 ;
                        001D  3859 ;       THIS TEST CHECKS THE DPM BRANCHING LOGIC USED BY THE
                        001D  3860 ;       "BUT/CCBR1.CCBR0.IR0" MICRO ORDER.
                        001D  3861 ;
                        001D  3862 ;
                        001D  3863 ; TEST ALGORITHM:
                        001D  3864 ;
                        001D  3865 ;       1.  CREATE A TEST LOOP OF 2
                        001D  3866 ;       2.  MODIFY AND LOAD MICRO CODE FOR EACH LOOP
                        001D  3867 ;       3.  SAVE THE LOOP COUNT IN THE WDREG
                        001D  3868 ;       4.  EXECUTE DCS PROGRAM
                        001D  3869 ;               a.  DISABLE ^P INTERRUPTS
                        001D  3870 ;               b.  MOVE A TEST PATTERN TO THE Q REGISTER
                        001D  3871 ;               c.  PERFORM AN ADD BETWEEN Q AND RTEMP0 WHILE SETTING THE
                        001D  3872 ;                       ALU STATE FLAGS
                        001D  3873 ;               d.  USE THE LOOP COUNT TO FORM A CACHE ADDRESS
                        001D  3874 ;               e.  LOAD THIS ADDRESS INTO THE PC TO FORCE PREFETCH
                        001D  3875 ;               f.  PERFORM IRD1TEST TO LOAD THE INSTRUCTION REGISTER
                        001D  3876 ;               g.  DO THE BRANCH
                        001D  3877 ;       5.  TEST THE CS ADDRESS FOR THE CORRECT BRANCH ADDRESS
                        001D  3878 ;       6.  IF NO ERROR GO TO STEP 2 FOR THE NEXT LOOP
                        001D  3879 ;
                        001D  3880 ;
                        001D  3881 ; TEST PATTERNS:
                        001D  3882 ;
                        001D  3883 ;       ITERATION          Q REG + RTEMP0          CCBR1   CCBR0   IR0
                        001D  3884 ;           1           00000080 + 00000000         1       0       0
                        001D  3885 ;           2           00000000 + 00000000         0       1       1
                        001D  3886 ;
                        001D  3887 ;
                        001D  3888 ; LOGIC DESCRIPTION:
                        001D  3889 ;
                        001D  3890 ;       THIS TEST CHECKS THE DISCRETE LOGIC ON DPM WHICH GATES THE CCBR1,
                        001D  3891 ;       CCBR0, AND IR0 SIGNALS ONTO CS ADDRESS BITS <2:0>.
                        001D  3892 ;
                        001D  3893 ;
                        001D  3894 ; ASSUMPTIONS:
                        001D  3895 ;
                        001D  3896 ;       THIS TEST ASSUMES THAT THE CONDITION CODE AND PREFETCH LOGIC
                        001D  3897 ;       ARE OPERATIONAL.
                        001D  3898 ;
                        001D  3899 ;
                        001D  3900 ; ERROR DESCRIPTION:
                        001D  3901 ;
                        001D  3902 ;       EXPECTED CS ADDRESS
                        001D  3903 ;       RECEIVED CS ADDRESS
                        001D  3904 ;       LOOP COUNT I
                        001D  3905 ;
                        001D  3906 ;--
                        001D  3907 ;*************************************************************************
```

```
                                001D  3908
                                001D  3909
                                001D  3910
                                001D  3911          INITIALIZE                              ;INIT THE CPU
                                001F  3912          LOOP            I,1,2                   ;TEST LOOP OF 2
                                0026  3913          LDFIELD         1$,I,B,2$,34,39         ;MODIFY U-CODE FOR THIS LOOP
                                0037  3914          LOADDCS         2$,,03,1                ;LOAD U-CODE INTO DCS
                                003E  3915          SAVEINDEX       I,5$                    ;SAVE LOOP COUNT
                                0043  3916          LOADREG         WDREG,5$,,L             ;PUT LOOP COUNT INTO WDREG
                                004B  3917          ERRLOOP                                 ;ERROR LOOP POINT
                                004D  3918          FETCH           0                       ;START DCS PROGRAM
                                0052  3919          BRSTCLK         <^X0B>                  ;END DCS PROGRAM
                                0056  3920          CMPREGMSK       CSTRP,3$,I,4$,,W,       ;TEST CS ADDRESS
                                0062  3921          IFERROR                                 ;ADDRESS INCORRECT
                                0067  3922          ENDLOOP         I                       ;END TEST LOOP
                                006A  3923          SKIP                                    ;GO TO NEXT TEST
                                0071  3924
                                0071  3925
                                0071  3926
                                0071  3927 BEGIN_TEST_DATA
                                0071
                                0071        ;-------------------------------------------------------------------------
                                0071  3928
                             00 01  0071  3929 1$:      .BYTE   01,00                           ;RSRC FIELD
                                0073  3930 2$:
U 03, 7700,C800,5BE5,0304,0470,1804 0073  3931 ;% 03   Q_R[TEMP1]                             ;TEST DATA TO Q REGISTER
                             1805  007E  3935 3$:      .WORD   ^X1805                          ;RESULTS TABLE
                             1802  0080  3936          .WORD   ^X1802
                             3FFF  0082  3937 4$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                         00000000  0084  3938 5$:      .LONG   ^X00000000                      ;LOOP COUNT STORAGE
                                0088  3939
                                0088  3940
                                0088  3941
                                0088  3942 BEGIN_CPU_DATA
                                0002  3943
                                0002  3944 DCS_DATA
                                0001  3945
U 00, 7700,C800,0364,0300,0470,1801 0001  3946 ;% 00   NOP
U 01, 7700,4800,5BE4,030C,2470,1802 000C  3950 ;% 01   WCTRL/LOADCRAR,WB_R[TEMP3]              ;DISABLE ^P INTERRUPT
U 02, 7700,0800,5BE4,0310,2C70,1803 0017  3954 ;% 02   WCTRL/CONWRITE,WB_R[TEMP4]              ;...
U 03, 7700,C800,5BE5,0304,0470,1804 0022  3958 ;% 03   Q_R[TEMP1]                             ;TEST DATA TO Q REGISTER
U 04, 7701,8800,0390,0001,8C70,1805 002D  3962 ;% 04   SIZE[BYTE],ALU/A+B+CI,MUX/R.Q,RSRC/TEMP0,CCMISC/ALUS_SIGND.CCBR_ALUS
                                0038  3966                                                  ;SET ALU FLAGS ACCORDING TO ADD
U 05, 5700,4840,0364,0308,0470,1806 0038  3967 ;% 05   R[TEMP2]_RDM                          ;LOOP COUNT TO RTEMP2
U 06, 7700,5800,EF64,0301,0470,1807 0043  3971 ;% 06   PL_[2]                                ;PLATCH GETS 2
U 07, 7700,C800,8B71,0308,0470,1808 004E  3975 ;% 07   Q_R[TEMP2].RL.P                       ;FORM CACHE ADDRESS
U 08, 7700,C800,03A4,03D8,4870,1809 0059  3979 ;% 08   WCTRL/PC_WB,WB_Q                       ;FORCE PREFETCH
U 09, 7700,4800,0364,1700,0470,180A 0064  3983 ;% 09   IRD1TEST                              ;LOAD IR REGISTER
U 0A, 7700,4800,0364,D701,8470,1800 006F  3987 ;% 0A   BUT/CCBR1.CCBR0.IR0,CC/NOP.CCBR_ALUS,NEXT/1800 ;BRANCH ON FLAGS AND IR0
U 0B, 7300,4800,0364,0300,0470,180C 007A  3991 ;% 0B   NOP,STOP.CLOCK                         ;STOP THE CPU CLOCK
                                0085  3995
                                0085  3996 RTEMP_DATA
                                0001  3997
                         00000000  0001  3998          .LONG   ^X00000000                      ;TEST DATA
                         00000080  0005  3999          .LONG   ^X00000080
```

```
00000000  0009  4000          .LONG   ^X00000000                      ;LOOP COUNT STORAGE
00C00000  000D  4001          .LONG   ^X00C00000                      ;DISABLE ^P INTERRUPT
00400000  0011  4002          .LONG   ^X00400000                      ;...
          0015  4003
          0015  4004 CACHE_DATA
          0001  4005
00000000  0001  4006          .LONG   ^X00000000                      ;INSTRUCTION TABLE
00000001  0005  4007          .LONG   ^X00000001
00000000  0009  4008          .LONG   ^X00000000
00000000  000D  4009          .LONG   ^X00000000
          0011  4010
          0011  4011 END_CPU_DATA
          0088  4012
          0088  4013
          0088  4014 END_TEST_DATA
          0088
          0088       ;-------------------------------------------------------------------------
          0088  4015 ENDTEST
```

```
0025        .SBTTL  "TEST   0A0     BUT ON WMUX EQUAL/NOT EQUAL ZERO
0025  4017 ;*******************************************************************************
0025  4018 ;++
0025  4019 ;
0025  4020 ; FUNCTIONAL DESCRIPTION:
0025  4021 ;
0025  4022 ;       THIS TEST CHECKS THE DPM LOGIC WHICH GATES THE WMUX EQUAL ZERO STATUS
0025  4023 ;       ONTO CS ADDRESS BIT 0.
0025  4024 ;
0025  4025 ; TEST ALGORITHM:
0025  4026 ;
0025  4027 ;       1.  LOAD MICRO CODE FOR THE SUBTEST
0025  4028 ;       2.  CREATE A TEST LOOP OF I
0025  4029 ;       3.  LOAD A TEST PATTERN INTO THE WDREG
0025  4030 ;       4.  EXECUTE DCS PROGRAM
0025  4031 ;               a.  MOVE TEST PATTERN TO R[TEMP0]
0025  4032 ;               b.  READ TEST PATTERN TO THE WBUS WHILE BRANCHING ON WMUX
0025  4033 ;                       CONDITION.
0025  4034 ;       5.  TEST THE CS ADDRESS FOR THE CORRECT BRANCH
0025  4035 ;       6.  IF NO ERROR GO TO STEP 3 FOR THE REMAINING TEST PATTERNS
0025  4036 ;
0025  4037 ;
0025  4038 ; TEST PATTERNS:
0025  4039 ;
0025  4040 ;       SUBTEST #1
0025  4041 ;               ITERATION       TEST PATTERN    CS ADDRESS
0025  4042 ;                   1             00000000        1801
0025  4043 ;                   2             000000FF        1800
0025  4044 ;                   3             0000FF00        1800
0025  4045 ;                   4             00FF0000        1800
0025  4046 ;                   5             FF000000        1800
0025  4047 ;
0025  4048 ;       SUBTEST #2
0025  4049 ;               ITERATION       TEST PATTERN    CS ADDRESS
0025  4050 ;                   1             00000000        1800
0025  4051 ;                   2             000000FF        1801
0025  4052 ;
0025  4053 ;
0025  4054 ; LOGIC DESCRIPTION:
0025  4055 ;
0025  4056 ;       THIS TEST CHECKS THE DISCRETE BRANCHING LOGIC ON DPM WHICH GATES
0025  4057 ;       THE WMUX EQUAL ZERO CONDITION ONTO CS ADDRESS BIT <0>.  THE MICRO
0025  4058 ;       ORDERS USED ARE: BUT/WX.EQ.0 AND BUT/WMUX.NE.0
0025  4059 ;
0025  4060 ;
0025  4061 ; ASSUMPTIONS:
0025  4062 ;
0025  4063 ;       THIS TEST ASSUMES THE WMUXZ SIGNALS HAVE BEEN TESTED IN PREVIOUS
0025  4064 ;       TESTS.
0025  4065 ;
0025  4066 ;
0025  4067 ; ERROR DESCRIPTION:
0025  4068 ;
0025  4069 ;       EXPECTED CS ADDRESS
0025  4070 ;       RECEIVED CS ADDRESS
```

```
                     0025  4071 ;       LOOP COUNT I
                     0025  4072 ;
                     0025  4073 ;--
                     0025  4074 ;*******************************************************************************
                     0025  4075 ;///////////////////////////////////////////////////////////////////////////////
                     0025  4076 ;+
                     0025  4077 ;SUBTEST #1
                     0025  4078 ;
                     0025  4079 ;       TEST BUT/WX.EQ.0
                     0025  4080 ;-
                     0025  4081         SUBTEST                                         ;SUBTEST #1
                     0028
                     0028       ;///////////////////////////////////////////////////////////////////////////////
                     0028  4082         INITIALIZE                                      ;INIT THE CPU
                     002A  4083         LOADDCS         5$,,02,1                        ;MODIFY MICRO CODE FOR SUBTEST
                     0031  4084         LOOP            I,1,5                           ;CREATE TEST LOOP OF 5
                     0038  4085         LOADREG         WDREG,1$,I,L                    ;TEST PATTERN TO WDREG
                     0040  4086         ERRLOOP                                         ;ERROR LOOP POINT
                     0042  4087         FETCH           0                               ;START DCS PROGRAM
                     0047  4088         BRSTCLK         3                               ;END DCS PROGRAM
                     004B  4089         CMPREGMSK       CSTRP,2$,I,4$,,W,               ;TEST CS ADDRESS
                     0057  4090         IFERROR                                         ;ADDRESS INCORRECT
                     005C  4091         ENDLOOP         I                               ;END TEST LOOP
                     005F  4092
                     005F  4093
                     005F  4094 ;///////////////////////////////////////////////////////////////////////////////
                     005F  4095 ;+
                     005F  4096 ;SUBTEST #2
                     005F  4097 ;
                     005F  4098 ;       TEST BUT/WX.NE.0
                     005F  4099 ;-
                     005F  4100         SUBTEST                                         ;SUBTEST #2
                     0062
                     0062       ;///////////////////////////////////////////////////////////////////////////////
                     0062  4101         INITIALIZE                                      ;INIT THE CPU
                     0064  4102         LOADDCS         6$,,02,1                        ;MODIFY MICRO CODE FOR SUBTEST
                     006B  4103         LOOP            I,1,2                           ;CREATE TEST LOOP OF 2
                     0072  4104         LOADREG         WDREG,1$,I,L                    ;LOAD TEST PATTERN INTO WDREG
                     007A  4105         ERRLOOP                                         ;ERROR LOOP POINT
                     007C  4106         FETCH           0                               ;START DCS PROGRAM
                     0081  4107         BRSTCLK         3                               ;END DCS PROGRAM
                     0085  4108         CMPREGMSK       CSTRP,3$,I,4$,,W,               ;TEST CS ADDRESS
                     0091  4109         IFERROR                                         ;ADDRESS INCORRECT
                     0096  4110         ENDLOOP         I                               ;END TEST LOOP
                     0099  4111         SKIP                                            ;GO TO NEXT TEST
                     00A0  4112
                     00A0  4113
                     00A0  4114 BEGIN_TEST_DATA
                     00A0
                     00A0       ;-------------------------------------------------------------------------------
                     00A0  4115
00000000             00A0  4116 1$:      .LONG   ^X00000000                              ;TEST PATTERNS
000000FF             00A4  4117          .LONG   ^X000000FF
0000FF00             00A8  4118          .LONG   ^X0000FF00
00FF0000             00AC  4119          .LONG   ^X00FF0000
```

```
                    FF000000  00B0  4120          .LONG   ^XFF000000
                        1801  00B4  4121 2$:      .WORD   ^X1801                          ;RESULTS TABLE
                        1800  00B6  4122          .WORD   ^X1800
                        1800  00B8  4123          .WORD   ^X1800
                        1800  00BA  4124          .WORD   ^X1800
                        1800  00BC  4125          .WORD   ^X1800
                        1800  00BE  4126 3$:      .WORD   ^X1800
                        1801  00C0  4127          .WORD   ^X1801
                        3FFF  00C2  4128 4$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                              00C4  4129 5$:                                            ;MICRO CODE SUBTEST #1
U 02, 7700,4800,5BE4,A300,0470,1800   00C4  4130 ;% 02   WB_R[TEMP0],BUT/WX.EQ.0,NEXT/1800  ;BUT ON EQUAL 0
                              00CF  4134 6$:                                            ;MICRO CODE SUBTEST #2
U 02, 7700,0800,5BE4,A700,0470,1800   00CF  4135 ;% 02   WB_R[TEMP0],BUT/WX.NE.0,NEXT/1800  ;BUT ON NOT EQUAL 0
                              00DA  4139
                              00DA  4140
                              00DA  4141
                              00DA  4142 BEGIN_CPU_DATA
                              0002  4143
                              0002  4144 DCS_DATA
                              0001  4145
U 00, 7700,C800,0364,0300,0470,1801   0001  4146 ;% 00   NOP
U 01, 5700,8840,0364,0300,0470,1802   000C  4150 ;% 01   R[TEMP0]_RDM                     ;TEST PATTERN TO RTEMP0
U 02, 7700,4800,5BE4,A300,0470,1800   0017  4154 ;% 02   WB_R[TEMP0],BUT/WX.EQ.0,NEXT/1800  ;TEST PATTERN THROUGH WMUX
U 03, 7300,C800,0364,0300,0470,1804   0022  4158 ;% 03   NOP,STOP.CLOCK                   ;STOP THE CPU CLOCK
                              002D  4162
                              002D  4163 END_CPU_DATA
                              00DA  4164
                              00DA  4165
                              00DA  4166 END_TEST_DATA
                              00DA
                              00DA        ;-----------------------------------------------------------------------------
                              00DA  4167 ENDTEST
```

```
0026        .SBTTL  "TEST   0A1     Rotator Test 50 SRK Status Test 1
0026  4169 ;*******************************************************************************
0026  4170 ;++
0026  4171 ;
0026  4172 ; FUNCTIONAL DESCRIPTION
0026  4173 ;
0026  4174 ;       This test examines the logic that implements the BUT/SRKSTA microbranch
0026  4175 ;       on SLATCH and PLATCH data. The BUT/SRKSTA microbranch is performed by
0026  4176 ;       placing two bits called SRKSTA<1:0> onto the Control Store Address Bus
0026  4177 ;       (CSADDR) bits <1:0>.
0026  4178 ;
0026  4179 ; TEST ALGORITHM
0026  4180 ;
0026  4181 ;       This test is implemented in six subtests. Each subtest examines one
0026  4182 ;       type of SLATCH and PLATCH SRK status microbranch. The microbranches
0026  4183 ;       performed by these subtests are:
0026  4184 ;
0026  4185 ;       Subtest 1 SRKSTA<1> = SL.EQ.0 and SRKSTA<0> = (PL<4:0>+SL).GT.32
0026  4186 ;       Subtest 2 SRKSTA<1> = SL.EQ.0 and SRKSTA<0> = PL<5>
0026  4187 ;       Subtest 3 SRKSTA<1> = 0 and SRKSTA<0> = PL<5>
0026  4188 ;       Subtest 4 SRKSTA<1> = PL<4:0>.EQ.0 and SRKSTA<0> = PL<5>
0026  4189 ;       Subtest 5 SRKSTA<1> = PL<4:0>.EQ.0 and SRKSTA<0> = 0
0026  4190 ;       Subtest 6 SRKSTA<1> = SL.EQ.0 and SRKSTA<0> = undefined
0026  4191 ;
0026  4192 ;       Together they provide a full test of all the SLATCH and PLATCH SRK
0026  4193 ;       status microbranches. The algorithm for each is specified below:
0026  4194 ;
0026  4195 ;       Subtests 1 - 6
0026  4196 ;
0026  4197 ;        1. Initialize the CPU. (Subtest 1 only)
0026  4198 ;        2. Start signature analysis.
0026  4199 ;        3. Prepare a loop on I for all the ROT microorders that can branch
0026  4200 ;           on the particular S and P latch function under test.
0026  4201 ;        4. Load the ROT field of the test microinstruction for the function
0026  4202 ;           to be tested for this iteration of the I loop.
0026  4203 ;        5. Load the test microinstruction into DCS at address 1807.
0026  4204 ;        6. Prepare a loop on J for the number of S and P latch data patterns
0026  4205 ;           needed to check the branch under test.
0026  4206 ;        7. Load the LONLIT field of a LONLIT_[0] microinstruction with a
0026  4207 ;           P latch data pattern from a table indexed by J.
0026  4208 ;        8. Load the LONLIT field of a LONLIT_[0] microinstruction with an
0026  4209 ;           S latch data pattern from a table indexed by J.
0026  4210 ;        9. Load the microword prepared in step 7 into DCS at address 1801.
0026  4211 ;       10. Load the microword prepared in step 8 into DCS at address 1804.
0026  4212 ;       11. Loop on error from this point.
0026  4213 ;       12. Execute the microcode to do the following:
0026  4214 ;                1. Load the LONLIT register with a PLATCH data pattern.
0026  4215 ;                2. Write the LONLIT register to the rotator's PLATCH.
0026  4216 ;                3. Write the PLATCH into MTEMP0.
0026  4217 ;                4. Load the LONLIT register with an SLATCH data pattern.
0026  4218 ;                5. Write the LONLIT register to the rotator's SLATCH.
0026  4219 ;                6. Write the SLATCH into MTEMP1.
0026  4220 ;                7. Source zero onto the RBUS and MBUS and perform the rotator
0026  4221 ;                   and BUT functions that execute the microbranch to be tested.
0026  4222 ;       13. Mask out bit 13 (and bit 0 for Subtest 6 only) of the microaddress
```

```
0026  4223 ;           in the Control Store Address Trap Register (CSTRP) and then compare
0026  4224 ;           it against the expected address from a table indexed by J.
0026  4225 ;       14. If the two do not match signal an error.
0026  4226 ;       15. Continue the loop on J for the remaining S and P latch data.
0026  4227 ;       16. Continue the loop on I for the remaining rotator functions.
0026  4228 ;       17. Stop signature analysis.
0026  4229 ;       18. End of test, skip to the next test. (Subtest 6 only)
0026  4230 ;
0026  4231 ; TEST DATA
0026  4232 ;
0026  4233 ;       The tables given below show the contents of the ROT field of the micro-
0026  4234 ;       word based on the index I and the contents of the PLATCH, SLATCH, and
0026  4235 ;       CS ADDRESS BUS based on the index J for each subtest. All data are
0026  4236 ;       given in hex unless noted otherwise.
0026  4237 ;
0026  4238 ;       Subtest 1
0026  4239 ;
0026  4240 ;            INDEX I    ROTATOR FUNCTION
0026  4241 ;
0026  4242 ;               01      00 = ROT/XZ.MR
0026  4243 ;               02      01 = ROT/XZ.MM
0026  4244 ;               03      02 = ROT/XZ.RR
0026  4245 ;               04      04 = ROT/RR.MR.P
0026  4246 ;               05      05 = ROT/RR.MM.P
0026  4247 ;               06      06 = ROT/RR.RR.P
0026  4248 ;               07      24 = ROT/RR.MR.PS
0026  4249 ;               08      25 = ROT/RR.MM.PS
0026  4250 ;               09      26 = ROT/RR.RR.PS
0026  4251 ;
0026  4252 ;          INDEX J  PLATCH  SLATCH CS ADDRESS
0026  4253 ;
0026  4254 ;               01      3E      01      1800
0026  4255 ;               02      3D      02      1800
0026  4256 ;               03      3B      04      1800
0026  4257 ;               04      37      08      1800
0026  4258 ;               05      2F      10      1800
0026  4259 ;               06      3F      01      1800
0026  4260 ;               07      1F      01      1800
0026  4261 ;               08      1F      02      1801
0026  4262 ;               09      1F      04      1801
0026  4263 ;               0A      1F      08      1801
0026  4264 ;               0B      1F      10      1801
0026  4265 ;               0C      3F      00      1802
0026  4266 ;               0D      1F      00      1802
0026  4267 ;
0026  4268 ;       Subtest 2
0026  4269 ;
0026  4270 ;            INDEX I    ROTATOR FUNCTION
0026  4271 ;
0026  4272 ;               01      21 = ROT/RL.MM.P
0026  4273 ;               02      22 = ROT/RL.RR.P
0026  4274 ;
0026  4275 ;          INDEX J  PLATCH  SLATCH CS ADDRESS
0026  4276 ;
0026  4277 ;               01      00      01      1800
```

```
0026  4278 ;                    02      01      02      1800
0026  4279 ;                    03      02      04      1800
0026  4280 ;                    04      04      08      1800
0026  4281 ;                    05      08      10      1800
0026  4282 ;                    06      10      20      1800
0026  4283 ;                    07      20      01      1801
0026  4284 ;                    08      21      02      1801
0026  4285 ;                    09      22      04      1801
0026  4286 ;                    0A      24      08      1801
0026  4287 ;                    0B      28      10      1801
0026  4288 ;                    0C      30      20      1801
0026  4289 ;                    0D      00      00      1802
0026  4290 ;                    0E      01      00      1802
0026  4291 ;                    0F      02      00      1802
0026  4292 ;                    10      04      00      1802
0026  4293 ;                    11      08      00      1802
0026  4294 ;                    12      10      00      1802
0026  4295 ;                    13      20      00      1803
0026  4296 ;                    14      21      00      1803
0026  4297 ;                    15      22      00      1803
0026  4298 ;                    16      24      00      1803
0026  4299 ;                    17      28      00      1803
0026  4300 ;                    18      30      00      1803
0026  4301 ;
0026  4302 ;          Subtest 3
0026  4303 ;
0026  4304 ;                INDEX I    ROTATOR FUNCTION
0026  4305 ;
0026  4306 ;                   01      03 = ROT/ASR.M.P
0026  4307 ;                   02      07 = ROT/RR.MR.S
0026  4308 ;
0026  4309 ;               INDEX J  PLATCH  SLATCH CS ADDRESS
0026  4310 ;
0026  4311 ;                   01      00      01      1800
0026  4312 ;                   02      01      01      1800
0026  4313 ;                   03      02      01      1800
0026  4314 ;                   04      04      01      1800
0026  4315 ;                   05      08      01      1800
0026  4316 ;                   06      10      01      1800
0026  4317 ;                   07      20      01      1801
0026  4318 ;                   08      21      01      1801
0026  4319 ;                   09      22      01      1801
0026  4320 ;                   0A      24      01      1801
0026  4321 ;                   0B      28      01      1801
0026  4322 ;                   0C      30      01      1801
0026  4323 ;
0026  4324 ;          Subtest 4
0026  4325 ;
0026  4326 ;                INDEX I    ROTATOR FUNCTION
0026  4327 ;
0026  4328 ;                   01      28 = ROT/ASL.R.P
0026  4329 ;                   02      29 = ROT/ASL.M.P
0026  4330 ;                   03      2A = ROT/ASR.M.-P
0026  4331 ;
0026  4332 ;               INDEX J  PLATCH  SLATCH CS ADDRESS
```

```
0026  4333 ;
0026  4334 ;                01      01      01      1800
0026  4335 ;                02      02      01      1800
0026  4336 ;                03      04      01      1800
0026  4337 ;                04      08      01      1800
0026  4338 ;                05      10      01      1800
0026  4339 ;                06      21      01      1801
0026  4340 ;                07      22      01      1801
0026  4341 ;                08      24      01      1801
0026  4342 ;                09      28      01      1801
0026  4343 ;                0A      30      01      1801
0026  4344 ;                0B      00      01      1802
0026  4345 ;                0C      20      01      1803
0026  4346 ;
0026  4347 ;       Subtest 5
0026  4348 ;
0026  4349 ;            INDEX I    ROTATOR FUNCTION
0026  4350 ;
0026  4351 ;               01      2B = ROT/ZLITPL
0026  4352 ;
0026  4353 ;            INDEX J  PLATCH  SLATCH CS ADDRESS
0026  4354 ;
0026  4355 ;               01      01      01      1800
0026  4356 ;               02      02      01      1800
0026  4357 ;               03      04      01      1800
0026  4358 ;               04      08      01      1800
0026  4359 ;               05      10      01      1800
0026  4360 ;               06      20      01      1802
0026  4361 ;
0026  4362 ;       Subtest 6
0026  4363 ;
0026  4364 ;            INDEX I    ROTATOR FUNCTION
0026  4365 ;
0026  4366 ;               01      20 = ROT/RL.RM.PS
0026  4367 ;
0026  4368 ;            INDEX J  PLATCH  SLATCH  CS ADDRESS
0026  4369 ;
0026  4370 ;               01      01      01      180oooox
0026  4371 ;               02      01      02      180oooox
0026  4372 ;               03      01      04      180oooox
0026  4373 ;               04      01      08      180oooox
0026  4374 ;               05      01      10      180oooox
0026  4375 ;               06      01      20      180oooox
0026  4376 ;               07      01      00      180oo1x
0026  4377 ;
0026  4378 ;       Note - The letters in CS ADDRESS indicate single bits.
0026  4379 ;              The x's are don't cares.
0026  4380 ;
0026  4381 ; LOGIC DESCRIPTION
0026  4382 ;
0026  4383 ;       A combination of discrete logic on the DPM module and logic internal to
0026  4384 ;       the SRK gate array is used to implement the SRK status microbranch.
0026  4385 ;       The SRK gate array contains the S and P latches and also receives the
0026  4386 ;       ROT field of the microword. This enables it to generate the signals
0026  4387 ;       SRK ST1 and SRK ST0 based on the contents of the S and P latches and
```

```
                        4388 ;       ROT field of the microword. These signals are passed to two 74S151
                        4389 ;       8 to 1 multiplexers that can drive bits 0 and 1 of the CS ADDRESS BUS.
                        4390 ;       The mux that receives SRK ST1 will select this signal and enable its
                        4391 ;       output whenever it receives a BUT/SRKSTA microbranch. The mux that
                        4392 ;       receives SRK ST0 will select this signal and enable its output whenever
                        4393 ;       it receives a BUT/SRKSTA or BUT/CCBR0.SRKSTA0 microbranch. For this
                        4394 ;       test only BUT/SRKSTA is used.
                        4395 ;
                        4396 ; ASSUMPTIONS
                        4397 ;
                        4398 ;       This test assumes the following DPM logic has been tested and is
                        4399 ;       working; the RBUS, MBUS and WBUS; the LONLIT register; the ALPCTL
                        4400 ;       microorders WX_R.Q_D and WX_S.Q_0; the rotator's S and P latches; and
                        4401 ;       the dual port scratchpad registers TEMP0, TEMP1 and TEMP2.
                        4402 ;
                        4403 ; ERROR DESCRIPTION
                        4404 ;
                        4405 ;       On error this tell will call out the SRK gate array and will type out
                        4406 ;       the following information:
                        4407 ;
                        4408 ;       EXPECTED CONTROL STORE ADDRESS
                        4409 ;       RECEIVED CONTROL STORE ADDRESS
                        4410 ;       INDEX I (ROTATOR FUNCTION UNDER TEST)
                        4411 ;       INDEX J (INDEX TO THE SLATCH AND PLATCH DATA TABLES)
                        4412 ;       RTEMP0 (PLATCH CONTENTS)
                        4413 ;       RTEMP1 (SLATCH CONTENTS)
                        4414 ;
                        4415 ;--
                        4416 ;*******************************************************************************
                        4417
                        4418         SUBTEST         1
                        4419
                             ;////////////////////////////////////////////////////////////////////////////
                        4420 ;+
                        4421 ; This subtest examines the BUT/SRKSTA function for SRKSTA<1> = SL.EQ.0 and
                        4422 ; SRKSTA<0> = (PL<4:0>+SL).GT.32
                        4423 ;-
0029                    4424         INITIALIZE                                  ; INITIALZE THE CPU
002B                    4425         LOOP            I,1,9                       ; LOOP FOR 9 ROT FUNCTIONS
0032                    4426         LDFIELD         4$,I,B,102$,58,63           ; LOADS ROT FIELD
0043                    4427         LOADDCS         102$,,7,1                   ; LOADS TEST INSTRUCTION
004A                    4428         LOOP            J,1,13                      ; LOOP FOR 13 TEST PATTERNS
0051                    4429         LDFIELD         1$,J,B,100$,31,62,LL        ; LOAD PLATCH DATA
0062                    4430         LDFIELD         2$,J,B,101$,31,62,LL        ; LOAD SLATCH DATA
0073                    4431         LOADDCS         100$,,1,1                   ; LOAD PLATCH LOAD INSTR.
007A                    4432         LOADDCS         101$,,4,1                   ; LOAD SLATCH LOAD INSTR.
0081                    4433         ERRLOOP                                     ; LOOP ON ERROR FROM HERE
0083                    4434         FETCH           0                           ; EXECUTE THE TEST
0088                    4435         BRSTCLK         8                           ; ...
008C                    4436         CMPREGMSK       CSTRP,3$,J,5$,,W            ; IS THE CS ADDRESS CORRECT?
0098                    4437         IFERROR         2,<SRK>                     ; IF NOT SIGNAL ERROR
009E                    4438         ENDLOOP         J                           ; CONTINUE FOR ALL PATTERNS
00A1                    4439         ENDLOOP         I                           ; CONTINUE FOR ALL INSTRUCTIONS
00A4                    4440
00A4                    4441         SUBTEST         2
```

```
00A7
00A7            ;//////////////////////////////////////////////////////////////////////////////
00A7    4441    ;+
00A7    4442    ; This subtest examines the BUT/SRKSTA function SRKSTA<1> = SL.EQ.0 and
00A7    4443    ; SRKSTA<0> = PL<5>.
00A7    4444    ;-
00A7    4445            LOOP            I,1,2                   ; LOOP FOR 2 ROT FUNCTIONS
00AE    4446            LDFIELD         9$,I,B,102$,58,63       ; LOADS ROT FIELD
00BF    4447            LOADDCS         102$,,7,1               ; LOAD TEST INSTR. INTO DCS
00C6    4448            LOOP            J,1,24                  ; LOOP ON J FOR 24 PATTERNS
00CD    4449            LDFIELD         6$,J,B,100$,31,62,LL    ; LOAD PLATCH DATA
00DE    4450            LDFIELD         7$,J,B,101$,31,62,LL    ; LOAD SLATCH DATA
00EF    4451            LOADDCS         100$,,1,1               ; LOAD PLATCH LOAD INSTR.
00F6    4452            LOADDCS         101$,,4,1               ; LOAD SLATCH LOAD INSTR.
00FD    4453            ERRLOOP                                 ; LOOP ON ERROR FROM HERE
00FF    4454            FETCH           0                       ; EXECUTE THE TEST
0104    4455            BRSTCLK         8                       ; ...
0108    4456            CMPREGMSK       CSTRP,8$,J,5$,,W        ; IS THE CS ADDRESS CORRECT?
0114    4457            IFERROR         2,<SRK>                 ; IF NOT SIGNAL ERROR
011A    4458            ENDLOOP         J                       ; CONTINUE FOR ALL PATTERNS
011D    4459            ENDLOOP         I                       ; CONTINUE FOR ALL FUNCTIONS
0120    4460
0120    4461            SUBTEST         3
0123
0123            ;//////////////////////////////////////////////////////////////////////////////
0123    4462    ;+
0123    4463    ; This subtest examines the BUT/SRKSTA function SRKSTA<1> = 0 and
0123    4464    ; SRKSTA<0> = PL<5>.
0123    4465    ;-
0123    4466            LOOP            I,1,2                   ; LOOP FOR 2 ROT FUNCTIONS
012A    4467            LDFIELD         13$,I,B,102$,58,63      ; LOADS ROT FIELD
013B    4468            LOADDCS         102$,,7,1               ; LOAD TEST INSTR. INTO DCS
0142    4469            LOOP            J,1,12                  ; LOOP ON J FOR 12 PATTERNS
0149    4470            LDFIELD         10$,J,B,100$,31,62,LL   ; LOAD PLATCH DATA
015A    4471            LDFIELD         11$,,B,101$,31,62,LL    ; LOAD SLATCH DATA
016B    4472            LOADDCS         100$,,1,1               ; LOAD PLATCH LOAD INSTR.
0172    4473            LOADDCS         101$,,4,1               ; LOAD SLATCH LOAD INSTR.
0179    4474            ERRLOOP                                 ; LOOP ON ERROR FROM HERE
017B    4475            FETCH           0                       ; EXECUTE THE TEST
0180    4476            BRSTCLK         8                       ; ...
0184    4477            CMPREGMSK       CSTRP,12$,J,5$,,W       ; IS THE CS ADDRESS CORRECT?
0190    4478            IFERROR         2,<SRK>                 ; IF NOT SIGNAL ERROR
0196    4479            ENDLOOP         J                       ; CONTINUE FOR ALL PATTERNS
0199    4480            ENDLOOP         I                       ; CONTINUE FOR ALL FUNCTIONS
019C    4481
019C    4482            SUBTEST         4
019F
019F            ;//////////////////////////////////////////////////////////////////////////////
019F    4483    ;+
019F    4484    ; This subtest examines the BUT/SRKSTA function SRKSTA<1> = PL<4:0>.EQ.0 and
019F    4485    ; SRKSTA<0> = PL<5>.
019F    4486    ;-
019F    4487            LOOP            I,1,3                   ; LOOP FOR 3 ROT FUNCTIONS
01A6    4488            LDFIELD         17$,I,B,102$,58,63      ; LOADS ROT FIELD
01B7    4489            LOADDCS         102$,,7,1               ; LOAD TEST INSTR. INTO DCS
```

```
01BE  4490          LOOP            J,1,12                  ; LOOP ON J FOR 12 PATTERNS
01C5  4491          LDFIELD         14$,J,B,100$,31,62,LL   ; LOAD PLATCH DATA
01D6  4492          LDFIELD         15$,,B,101$,31,62,LL    ; LOAD SLATCH DATA
01E7  4493          LOADDCS         100$,,1,1               ; LOAD PLATCH LOAD INSTR.
01EE  4494          LOADDCS         101$,,4,1               ; LOAD SLATCH LOAD INSTR.
01F5  4495          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
01F7  4496          FETCH           0                       ; EXECUTE THE TEST
01FC  4497          BRSTCLK         8                       ; ...
0200  4498          CMPREGMSK       CSTRP,16$,J,5$,,W       ; IS THE CS ADDRESS CORRECT?
020C  4499          IFERROR         2,<SRK>                 ; IF NOT SIGNAL ERROR
0212  4500          ENDLOOP         J                       ; CONTINUE FOR ALL PATTERNS
0215  4501          ENDLOOP         I                       ; CONTINUE FOR ALL FUNCTIONS
0218  4502
0218  4503          SUBTEST         5
021B
021B              ;////////////////////////////////////////////////////////////////////////////////
021B  4504  ;+
021B  4505  ; This subtest examines the BUT/SRKSTA function SRKSTA<1> = PL<4:0>.EQ.0 and
021B  4506  ; SRKSTA<0> = 0.
021B  4507  ;-
021B  4508          LOOP            I,1,1                   ; LOOP FOR 1 ROT FUNCTION
0222  4509          LDFIELD         21$,I,B,102$,58,63      ; LOADS ROT FIELD
0233  4510          LOADDCS         102$,,7,1               ; LOAD TEST INSTR. INTO DCS
023A  4511          LOOP            J,1,6                   ; LOOP ON J FOR 6 PATTERNS
0241  4512          LDFIELD         18$,J,B,100$,31,62,LL   ; LOAD PLATCH DATA
0252  4513          LDFIELD         19$,,B,101$,31,62,LL    ; LOAD SLATCH DATA
0263  4514          LOADDCS         100$,,1,1               ; LOAD PLATCH LOAD INSTR.
026A  4515          LOADDCS         101$,,4,1               ; LOAD SLATCH LOAD INSTR.
0271  4516          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
0273  4517          FETCH           0                       ; EXECUTE THE TEST
0278  4518          BRSTCLK         8                       ; ...
027C  4519          CMPREGMSK       CSTRP,20$,J,5$,,W       ; IS THE CS ADDRESS CORRECT?
0288  4520          IFERROR         2,<SRK>                 ; IF NOT SIGNAL ERROR
028E  4521          ENDLOOP         J                       ; CONTINUE FOR ALL PATTERNS
0291  4522          ENDLOOP         I                       ; END OF SINGLE FUNCTION
0294  4523
0294  4524          SUBTEST         6
0297
0297              ;////////////////////////////////////////////////////////////////////////////////
0297  4525  ;+
0297  4526  ; This subtest examines the BUT/SRKSTA function SRKSTA<1> = SL.EQ.0
0297  4527  ; (SRKSTA<0> is undefined).
0297  4528  ;-
0297  4529          LOOP            I,1,1                   ; LOOP FOR 1 ROT FUNCTION
029E  4530          LDFIELD         25$,I,B,102$,58,63      ; LOADS ROT FIELD
02AF  4531          LOADDCS         102$,,7,1               ; LOAD TEST INSTR. INTO DCS
02B6  4532          LOOP            J,1,7                   ; LOOP ON J FOR 7 PATTERNS
02BD  4533          LDFIELD         22$,,B,100$,31,62,LL    ; LOAD PLATCH DATA
02CE  4534          LDFIELD         23$,J,B,101$,31,62,LL   ; LOAD SLATCH DATA
02DF  4535          LOADDCS         100$,,1,1               ; LOAD PLATCH LOAD INSTR.
02E6  4536          LOADDCS         101$,,4,1               ; LOAD SLATCH LOAD INSTR.
02ED  4537          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
02EF  4538          FETCH           0                       ; EXECUTE THE TEST
02F4  4539          BRSTCLK         8                       ; ...
02F8  4540          CMPREGMSK       CSTRP,24$,J,26$,,W      ; IS THE CS ADDRESS CORRECT?
```

```
                        0304  4541          IFERROR         2,<SRK>                 ; IF NOT SIGNAL ERROR
                        030A  4542          ENDLOOP         J                       ; CONTINUE FOR ALL PATTERNS
                        030D  4543          ENDLOOP         I                       ; END LOOP FOR ROT FUNCTION
                        0310  4544          SKIP                                    ; END OF TEST
                        0317  4545
                        0317  4546 BEGIN_TEST_DATA
                        0317
                        0317        ;-------------------------------------------------------------------------
                        0317  4547 ;+
                        0317  4548 ; The subtest data given below is arranged by tables in the following manner:
                        0317  4549 ;
                        0317  4550 ;       PLATCH data table
                        0317  4551 ;       SLATCH data table
                        0317  4552 ;       Expected CS address table
                        0317  4553 ;       ROT field microorder table
                        0317  4554 ;-
                        0317  4555
                        0317  4556 ; Subtest 1
                        0317  4557
1F 3F 2F 37 3B 3D 3E    0317  4558 1$:     .BYTE   ^X3E,^X3D,^X3B,^X37,^X2F,^X3F,^X1F      ; SRKSTA<0> = 0
      1F 1F 1F 1F       031E  4559          .BYTE   ^X1F,^X1F,^X1F,^X1F                     ; SRKSTA<0> = 1
            1F 3F       0322  4560          .BYTE   ^X3F,^X1F                               ; SRKSTA<0> = 0
                        0324  4561
01 01 10 08 04 02 01    0324  4562 2$:     .BYTE   ^X01,^X02,^X04,^X08,^X10,^X01,^X01      ; SRKSTA<1:0> = 00
      10 08 04 02       032B  4563          .BYTE   ^X02,^X04,^X08,^X10                     ; SRKSTA<1:0> = 01
            00 00       032F  4564          .BYTE   ^X00,^X00                               ; SRKSTA<1:0> = 10
                        0331  4565
1800 1800 1800 1800 1800 1800 1800  0331  4566 3$:     .WORD   ^X1800,^X1800,^X1800,^X1800,^X1800,^X1800,^X1800
      1801 1801 1801 1801   033F  4567          .WORD   ^X1801,^X1801,^X1801,^X1801
            1802 1802       0347  4568          .WORD   ^X1802,^X1802
                            034B  4569
                        00  034B  4570 4$:     .BYTE   ^X00    ; ROT/XZ.MR
                        01  034C  4571          .BYTE   ^X01    ; ROT/XZ.MM
                        02  034D  4572          .BYTE   ^X02    ; ROT/XZ.RR
                        04  034E  4573          .BYTE   ^X04    ; ROT/RR.MR.P
                        05  034F  4574          .BYTE   ^X05    ; ROT/RR.MM.P
                        06  0350  4575          .BYTE   ^X06    ; ROT/RR.RR.P
                        24  0351  4576          .BYTE   ^X24    ; ROT/RR.MR.PS
                        25  0352  4577          .BYTE   ^X25    ; ROT/RR.MM.PS
                        26  0353  4578          .BYTE   ^X26    ; ROT/RR.RR.PS
                            0354  4579
                            0354  4580 ; Comparison mask for Subtests 1 - 5.
                            0354  4581
                      3FFF  0354  4582 5$:     .WORD   ^X3FFF
                            0356  4583
                            0356  4584 ; Subtest 2
                            0356  4585
10 08 04 02 01 00           0356  4586 6$:     .BYTE   ^X00,^X01,^X02,^X04,^X08,^X10   ; SRKSTA<0> = 0
30 28 24 22 21 20           035C  4587          .BYTE   ^X20,^X21,^X22,^X24,^X28,^X30   ; SRKSTA<0> = 1
10 08 04 02 01 00           0362  4588          .BYTE   ^X00,^X01,^X02,^X04,^X08,^X10   ; SRKSTA<0> = 0
30 28 24 22 21 20           0368  4589          .BYTE   ^X20,^X21,^X22,^X24,^X28,^X30   ; SRKSTA<0> = 1
                            036E  4590
20 10 08 04 02 01           036E  4591 7$:     .BYTE   ^X01,^X02,^X04,^X08,^X10,^X20   ; SRKSTA<1> = 0
20 10 08 04 02 01           0374  4592          .BYTE   ^X01,^X02,^X04,^X08,^X10,^X20   ; SRKSTA<1> = 0
00 00 00 00 00 00           037A  4593          .BYTE   ^X00,^X00,^X00,^X00,^X00,^X00   ; SRKSTA<1> = 1
```

```
                    00 00 00 00 00 00   0380  4594          .BYTE   ^X00,^X00,^X00,^X00,^X00,^X00   ; SRKSTA<1> = 1
                                        0386  4595
      1800 1800 1800 1800 1800 1800     0386  4596 8$:      .WORD   ^X1800,^X1800,^X1800,^X1800,^X1800,^X1800
      1801 1801 1801 1801 1801 1801     0392  4597          .WORD   ^X1801,^X1801,^X1801,^X1801,^X1801,^X1801
      1802 1802 1802 1802 1802 1802     039E  4598          .WORD   ^X1802,^X1802,^X1802,^X1802,^X1802,^X1802
      1803 1803 1803 1803 1803 1803     03AA  4599          .WORD   ^X1803,^X1803,^X1803,^X1803,^X1803,^X1803
                                        03B6  4600
                              21        03B6  4601 9$:      .BYTE   ^X21    ; ROT/RL.MM.P
                              22        03B7  4602          .BYTE   ^X22    ; ROT/RL.RR.P
                                        03B8  4603
                                        03B8  4604 ; Subtest 3
                                        03B8  4605
                    10 08 04 02 01 00   03B8  4606 10$:     .BYTE   ^X00,^X01,^X02,^X04,^X08,^X10   ; SRKSTA<1:0> = 00
                    30 28 24 22 21 20   03BE  4607          .BYTE   ^X20,^X21,^X22,^X24,^X28,^X30   ; SRKSTA<1:0> = 01
                                        03C4  4608
                              01        03C4  4609 11$:     .BYTE   ^X01
                                        03C5  4610
      1800 1800 1800 1800 1800 1800     03C5  4611 12$:     .WORD   ^X1800,^X1800,^X1800,^X1800,^X1800,^X1800
      1801 1801 1801 1801 1801 1801     03D1  4612          .WORD   ^X1801,^X1801,^X1801,^X1801,^X1801,^X1801
                                        03DD  4613
                              03        03DD  4614 13$:     .BYTE   ^X03    ; ROT/ASR.M.P
                              07        03DE  4615          .BYTE   ^X07    ; ROT/RR.MR.S
                                        03DF  4616
                                        03DF  4617 ; Subtest 4
                                        03DF  4618
                       10 08 04 02 01   03DF  4619 14$:     .BYTE   ^X01,^X02,^X04,^X08,^X10        ; SRKSTA<1:0> = 00
                       30 28 24 22 21   03E4  4620          .BYTE   ^X21,^X22,^X24,^X28,^X30        ; SRKSTA<1:0> = 01
                                 00     03E9  4621          .BYTE   ^X00                            ; SRKSTA<1:0> = 10
                                 20     03EA  4622          .BYTE   ^X20                            ; SRKSTA<1:0> = 11
                                        03EB  4623
                                 01     03EB  4624 15$:     .BYTE   ^X01
                                        03EC  4625
            1800 1800 1800 1800 1800    03EC  4626 16$:     .WORD   ^X1800,^X1800,^X1800,^X1800,^X1800
            1801 1801 1801 1801 1801    03F6  4627          .WORD   ^X1801,^X1801,^X1801,^X1801,^X1801
                                1802    0400  4628          .WORD   ^X1802
                                1803    0402  4629          .WORD   ^X1803
                                        0404  4630
                                 28     0404  4631 17$:     .BYTE   ^X28    ; ROT/ASL.R.P
                                 29     0405  4632          .BYTE   ^X29    ; ROT/ASL.M.P
                                 2A     0406  4633          .BYTE   ^X2A    ; ROT/ASR.M.-P
                                        0407  4634
                                        0407  4635 ; Subtest 5
                                        0407  4636
                       10 08 04 02 01   0407  4637 18$:     .BYTE   ^X01,^X02,^X04,^X08,^X10        ; SRKSTA<1:0> = 00
                                 20     040C  4638          .BYTE   ^X20                            ; SRKSTA<1:0> = 10
                                        040D  4639
                                 01     040D  4640 19$:     .BYTE   ^X01
                                        040E  4641
            1800 1800 1800 1800 1800    040E  4642 20$:     .WORD   ^X1800,^X1800,^X1800,^X1800,^X1800
                                1802    0418  4643          .WORD   ^X1802
                                        041A  4644
                                 2B     041A  4645 21$:     .BYTE   ^X2B    ; ROT/ZLITPL
                                        041B  4646
                                        041B  4647 ; Subtest 6
                                        041B  4648
```

```
                                 01   041B  4649 22$:     .BYTE   ^X01
                                      041C  4650
                  20 10 08 04 02 01   041C  4651 23$:     .BYTE   ^X01,^X02,^X04,^X08,^X10,^X20   ; SRKSTA<1:0> = 0x
                                 00   0422  4652           .BYTE   ^X00                            ; SRKSTA<1:0> = 1x
                                      0423  4653
      1800 1800 1800 1800 1800 1800   0423  4654 24$:     .WORD   ^X1800,^X1800,^X1800,^X1800,^X1800,^X1800
                               1802   042F  4655           .WORD   ^X1802
                                      0431  4656
                                 20   0431  4657 25$:     .BYTE   ^X20    ; ROT/RL.RM.PS
                                      0432  4658
                                      0432  4659 ; Mask for address comparions done in Subtest 6.
                                      0432  4660
                               3FFE   0432  4661 26$:     .WORD   ^X3FFE
                                      0434  4662
                                      0434  4663 ;
                                      0434  4664 ; Reloadable test microwords.
                                      0434  4665 ;
                                      0434  4666
                                      0434  4667 100$:
U 01, 7700,7800,7FFF,FFFF,8470,1802   0434  4668 ;% 01   LONLIT_[0]
                                      043F  4672
                                      043F  4673 101$:
U 04, 7700,F800,7FFF,FFFF,8470,1805   043F  4674 ;% 04   LONLIT_[0]
                                      044A  4678
                                      044A  4679 102$:
U 07, 7700,0802,0364,DF08,0470,1800   044A  4680 ;% 07   RSRC/TEMP2,MSRC/TEMP2,ROT/XZ.MR,BUT/SRKSTA,NEXT/1800
                                      0455  4684
                                      0455  4685 BEGIN_CPU_DATA
                                      0002  4686
                                      0002  4687 DCS_DATA
                                      0001  4688
U 00, 7700,C800,0364,0300,0470,1801   0001  4689 ;% 00   NOP
U 01, 7700,7800,7FFF,FFFF,8470,1802   000C  4693 ;% 01   LONLIT_[0]
U 02, 7700,0800,BE47,03D4,0470,1803   0017  4697 ;% 02   RSRC/LONLIT,ALPCTL/WX_R.Q_D,ROT/SL.PL_WB
U 03, 7700,C860,B347,0300,0470,1804   0022  4701 ;% 03   ROT/PL,ALPCTL/WX_S.Q_0,MSRC/TEMP0,SPW/MLONG
U 04, 7700,F800,7FFF,FFFF,8470,1805   002D  4705 ;% 04   LONLIT_[0]
U 05, 7700,4800,B647,03D4,0470,1806   0038  4709 ;% 05   RSRC/LONLIT,ALPCTL/WX_R.Q_D,ROT/PL.SL_WB
U 06, 7700,C861,BB47,0300,0470,1807   0043  4713 ;% 06   ROT/SL,ALPCTL/WX_S.Q_0,MSRC/TEMP1,SPW/MLONG
U 07, 7700,0802,0364,DF08,0470,1800   004E  4717 ;% 07   RSRC/TEMP2,MSRC/TEMP2,ROT/XZ.MR,BUT/SRKSTA,NEXT/1800
U 08, 7300,4800,0364,0300,0470,1809   0059  4721 ;% 08   STOP.CLOCK
                                      0064  4725
                                      0064  4726 RTEMP_DATA
                                      0001  4727
                           00000000   0001  4728          .LONG   0       ; RTEMP0 = PLATCH CONTENTS
                           00000000   0005  4729          .LONG   0       ; RTEMP1 = SLATCH CONTENTS
                           00000000   0009  4730          .LONG   0       ; RTEMP2 = CONSTANT 0
                                      000D  4731
                                      000D  4732 END_CPU_DATA
                                      0455  4733
                                      0455  4734 END_TEST_DATA
                                      0455
                                      0455        ;-------------------------------------------------------------------------
                                      0455  4735
                                      0455  4736 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:   TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
0026        .SBTTL  "TEST   0A2     Rotator Test 51 SRK Status Test 2
0026  4738 ;*******************************************************************************
0026  4739 ;++
0026  4740 ;
0026  4741 ; FUNCTIONAL DESCRIPTION
0026  4742 ;
0026  4743 ;       This test examines the SRK logic that implements the BUT/SRKSTA micro-
0026  4744 ;       branches based on DSIZE, SBUS and WBUS data. The BUT/SRKSTA microbranch
0026  4745 ;       is performed by placing two bits called SRKSTA<1:0> onto the Control
0026  4746 ;       Store Address Bus (CSADDR) bits <1:0>.
0026  4747 ;
0026  4748 ; TEST ALGORITHM
0026  4749 ;
0026  4750 ;       This test has been divided into three subtests. Each subtest examines
0026  4751 ;       one of the three types of SRK status microbranches mentioned above.
0026  4752 ;       The microbranches performed by each subtest are:
0026  4753 ;
0026  4754 ;       Subtest 1 SRKSTA<1> = DSIZE<1> and SRKSTA<0> = DSIZE<0>
0026  4755 ;       Subtest 2 SRKSTA<1> = S<3:0>.NE.0 and SRKSTA<0> = S<3:0>.NE.(11,13)
0026  4756 ;       Subtest 3 SRKSTA<1> = WX<31:16>.NE.0 and SRKSTA<0> = WX<15:00>.NE.0
0026  4757 ;
0026  4758 ;       Subtest 1
0026  4759 ;
0026  4760 ;        1. Initialize the CPU.
0026  4761 ;        2. Start signature analysis.
0026  4762 ;        3. Prepare a loop on I for 8 rotator functions that perform DSIZE
0026  4763 ;           branching.
0026  4764 ;        4. Load the ROT field of the test microinstruction for the function
0026  4765 ;           to be tested for this iteration of the I loop.
0026  4766 ;        5. Prepare a loop on J for the 3 data types that can be specified in
0026  4767 ;           the DTYPE field of the test microinstruction.
0026  4768 ;        6. Load the DTYPE field of the test microinstruction for the DSIZE
0026  4769 ;           microbranch to be tested for this iteration of the J loop.
0026  4770 ;        7. Load the 3 test microinstructions into DCS starting at 1800.
0026  4771 ;        8. Loop on error from here.
0026  4772 ;        9. Execute the following microcode to perform a DSIZE microbranch:
0026  4773 ;               1. Put zeros on the RBUS, load DSIZE from the DTYPE field and
0026  4774 ;                  then perform the rotator and BUT functions that execute the
0026  4775 ;                  microbranch to be tested.
0026  4776 ;       10. Mask out bit 13 of the microaddress in the Control Store Address
0026  4777 ;           Trap Register (CSTRP) and then compare it against the expected
0026  4778 ;           address from a table indexed by J.
0026  4779 ;       11. If the two do not match signal an error.
0026  4780 ;       12. Continue the loop on J for the remaining DTYPE datums.
0026  4781 ;       13. Continue the loop on I for the remaining DSIZE microbranches.
0026  4782 ;       14. Stop signature analysis.
0026  4783 ;
0026  4784 ;       Subtest 2
0026  4785 ;
0026  4786 ;        1. Start signature analysis.
0026  4787 ;        2. Load DCS with four test microinstructions starting at address 1800.
0026  4788 ;        3. Prepare to loop on I for 24 MBUS to SBUS data patterns.
0026  4789 ;        4. Load the WDREG with an MBUS to SBUS test pattern from a table
0026  4790 ;           indexed by I.
0026  4791 ;        5. Loop on error from here.
```

```
0026  4792 ;        6. Execute the following test microcode:
0026  4793 ;              1. NOP to get the CPU clock up to speed.
0026  4794 ;              2. Load MTEMP0 with the pattern in WDREG.
0026  4795 ;              3. Put MTEMP0 onto the MBUS, execute a ROT/BCDSWP microorder,
0026  4796 ;                 and write the resulting SBUS contents to RTEMP1, while
0026  4797 ;                 performing the BUT/SRKSTA microbranch on an address of 1800.
0026  4798 ;              4. Stop the CPU clock to halt.
0026  4799 ;        7. Mask out bit 13 of the microaddress in the Control Store Address
0026  4800 ;           Trap Register (CSTRP) and then compare it against the expected
0026  4801 ;           address from a table indexed by I.
0026  4802 ;        8. If the two do not match, signal an error.
0026  4803 ;        9. Continue the loop on I for the remaining MBUS to SBUS patterns.
0026  4804 ;       10. Stop signature analysis.
0026  4805 ;
0026  4806 ;       Subtest 3
0026  4807 ;
0026  4808 ;        1. Start signature analysis.
0026  4809 ;        2. Prepare a loop on I for the 2 ROT microorders that can branch on
0026  4810 ;           the contents of WX<31:16> and WX<15:00>.
0026  4811 ;        3. Load the ROT field of the test microinstruction for the function
0026  4812 ;           to be tested by this iteration of the I loop.
0026  4813 ;        4. Load the four test microinstructions into DCS starting at 1800.
0026  4814 ;        5. Prepare to loop on J for 19 WX<31:00> test patterns.
0026  4815 ;        6. Load a WX<31:00> test pattern from a table indexed by J into the
0026  4816 ;           WDREG.
0026  4817 ;        7. Loop on error from here.
0026  4818 ;        8. Execute the following test microcode:
0026  4819 ;              1. NOP to get the CPU clock up to speed.
0026  4820 ;              2. Load RTEMP0 with the WX pattern in WDREG.
0026  4821 ;              3. Put RTEMP0 onto the RBUS, route it through the ALU to the
0026  4822 ;                 WBUS, perform the rotator function under test, write the
0026  4823 ;                 data on the WBUS to MTEMP1 and perform the BUT microbranch.
0026  4824 ;              4. Stop the CPU clock to halt.
0026  4825 ;        9. Mask out bit 13 of the microaddress in the Control Store Address
0026  4826 ;           Trap Register (CSTRP) and then compare it against the expected
0026  4827 ;           address from a table indexed by J.
0026  4828 ;       10. If the two do not match, signal an error.
0026  4829 ;       11. Continue the loop on J for the remaining WX test patterns.
0026  4830 ;       12. Continue the loop on I for the remaining rotator functions.
0026  4831 ;       13. Stop signature analysis.
0026  4832 ;       14. End of test, skip to the next test.
0026  4833 ;
0026  4834 ; TEST DATA
0026  4835 ;
0026  4836 ;       Subtest 1
0026  4837 ;
0026  4838 ;            INDEX I    ROTATOR FUNCTION
0026  4839 ;
0026  4840 ;               01      08 = ROT/RL.RM.4
0026  4841 ;               02      09 = ROT/RR.MR.4
0026  4842 ;               03      0A = ROT/RR.RR.SIZ
0026  4843 ;               04      0B = ROT/RR.MR.9
0026  4844 ;               05      0C = ROT/XZ.PTX
0026  4845 ;               06      0D = ROT/XZ.VPN
0026  4846 ;               07      0E = ROT/RR.MM.SIZ
```

```
0026  4847 ;                08      0F = ROT/GETNIB
0026  4848 ;
0026  4849 ;             INDEX J      DTYPE       CS ADDRESS
0026  4850 ;
0026  4851 ;                01      00 = BYTE       1800
0026  4852 ;                02      01 = WORD       1801
0026  4853 ;                03      02 = LONG       1802
0026  4854 ;
0026  4855 ;        Subtest 2
0026  4856 ;
0026  4857 ;             INDEX I      MBUS            SBUS        CS ADDRESS
0026  4858 ;
0026  4859 ;                01      A0AAAAAA        AAAAAAA0        1801
0026  4860 ;                02      50555555        55555550        1801
0026  4861 ;                03      C0CCCCCC        CCCCCCC0        1801
0026  4862 ;                04      F0F0F0F0        F0F0F0F0        1801
0026  4863 ;                05      00FF00FF        FF00FF00        1801
0026  4864 ;                06      0000FFFF        FFFF0000        1801
0026  4865 ;                07      ABAAAAAA        AAAAAAAB        1802
0026  4866 ;                08      5B555555        5555555B        1802
0026  4867 ;                09      3B333333        3333333B        1802
0026  4868 ;                0A      0B0F0F0F        0F0F0F0B        1802
0026  4869 ;                0B      FB00FF00        00FF00FB        1802
0026  4870 ;                0C      FBFF0000        0000FFFB        1802
0026  4871 ;                0D      ADAAAAAA        AAAAAAAD        1802
0026  4872 ;                0E      5D555555        5555555D        1802
0026  4873 ;                0F      3D333333        3333333D        1802
0026  4874 ;                10      0D0F0F0F        0F0F0F0D        1802
0026  4875 ;                11      FD00FF00        00FF00FD        1802
0026  4876 ;                12      FDFF0000        0000FFFD        1802
0026  4877 ;                13      AAAAAAAA        AAAAAAAA        1803
0026  4878 ;                14      55555555        55555555        1803
0026  4879 ;                15      33333333        33333333        1803
0026  4880 ;                16      0F0F0F0F        0F0F0F0F        1803
0026  4881 ;                17      FF00FF00        00FF00FF        1803
0026  4882 ;                18      FFFF0000        0000FFFF        1803
0026  4883 ;
0026  4884 ;        Subtest 3
0026  4885 ;
0026  4886 ;             INDEX I    ROTATOR FUNCTION
0026  4887 ;
0026  4888 ;                01      27 = ROT/PL_MSS
0026  4889 ;                02      23 = ROT/RL.RM.P
0026  4890 ;
0026  4891 ;             INDEX J      RBUS            WBUS        CS ADDRESS
0026  4892 ;
0026  4893 ;                01      00000000        00000000        1800
0026  4894 ;                02      0000AAAA        0000AAAA        1801
0026  4895 ;                03      00005555        00005555        1801
0026  4896 ;                04      00003333        00003333        1801
0026  4897 ;                05      00000F0F        00000F0F        1801
0026  4898 ;                06      000000FF        000000FF        1801
0026  4899 ;                07      0000FFFF        0000FFFF        1801
0026  4900 ;                08      AAAA0000        AAAA0000        1802
0026  4901 ;                09      55550000        55550000        1802
```

```
0026  4902 ;                0A      CCCC0000        CCCC0000        1802
0026  4903 ;                0B      F0F00000        F0F00000        1802
0026  4904 ;                0C      FF000000        FF000000        1802
0026  4905 ;                0D      FFFF0000        FFFF0000        1802
0026  4906 ;                0E      AAAAAAAA        AAAAAAAA        1803
0026  4907 ;                0F      55555555        55555555        1803
0026  4908 ;                10      33333333        33333333        1803
0026  4909 ;                11      0F0F0F0F        0F0F0F0F        1803
0026  4910 ;                12      00FF00FF        00FF00FF        1803
0026  4911 ;                13      0000FFFF        0000FFFF        1803
0026  4912 ;
0026  4913 ; LOGIC DESCRIPTION
0026  4914 ;
0026  4915 ;       A combination of discrete logic on the DPM module and logic internal to
0026  4916 ;       the SRK gate array is used to implement the SRK status microbranch.
0026  4917 ;       The SRK gate array receives the ROT field of the microword and the
0026  4918 ;       signals D SIZE<1:0>, SBUS<7:0> and WMUXZ<3:0>. This enables it to
0026  4919 ;       generate the signals SRK ST1 and SRK ST0 for each of the subtests in
0026  4920 ;       this test. These signals are passed to two 74S151 8 to 1 multiplexers
0026  4921 ;       that can drive bits 0 and 1 of the CS ADDRESS BUS. The mux that
0026  4922 ;       receives SRK ST1 will select this signal and enable its output whenever
0026  4923 ;       it receives a BUT/SRKSTA microbranch. The mux that receives SRK ST0
0026  4924 ;       will select this signal and enable its output whenever it receives a
0026  4925 ;       BUT/SRKSTA or BUT/CCBR0.SRKSTA0 microbranch. For this test only
0026  4926 ;       BUT/SRKSTA is used.
0026  4927 ;
0026  4928 ; ASSUMPTIONS
0026  4929 ;
0026  4930 ;       This test assumes the following DPM logic has been tested and is
0026  4931 ;       working; the RBUS, MBUS, SBUS and WBUS; the ALPCTL microorders WX_R.Q_D
0026  4932 ;       and WX_S.Q_0; the rotator's data path; and the dual port scratchpad
0026  4933 ;       registers TEMP0 and TEMP1.
0026  4934 ;
0026  4935 ; ERROR DESCRIPTION
0026  4936 ;
0026  4937 ;       On error this tell will call out the SRK gate array and will type out
0026  4938 ;       the following information:
0026  4939 ;
0026  4940 ;       Subtest 1
0026  4941 ;
0026  4942 ;       EXPECTED CONTROL STORE ADDRESS
0026  4943 ;       RECEIVED CONTROL STORE ADDRESS
0026  4944 ;       INDEX I (INDEX TO THE ROTATOR FUNCTION TABLE)
0026  4945 ;       INDEX J (INDEX TO THE DTYPE DATA TABLE)
0026  4946 ;
0026  4947 ;       Subtest 2
0026  4948 ;
0026  4949 ;       EXPECTED CONTROL STORE ADDRESS
0026  4950 ;       RECEIVED CONTROL STORE ADDRESS
0026  4951 ;       INDEX I (INDEX TO THE MBUS DATA TABLE)
0026  4952 ;       MTEMP0 (CONTAINS THE DRIVEN MBUS DATA PATTERN)
0026  4953 ;       RTEMP1 (CONTAINS THE RECEIVED MBUS-SBUS-WBUS DATA PATTERN)
0026  4954 ;
0026  4955 ;       Subtest 3
0026  4956 ;
```

```
                        4957 ;       EXPECTED CONTROL STORE ADDRESS
                        4958 ;       RECEIVED CONTROL STORE ADDRESS
                        4959 ;       INDEX I (INDEX TO THE ROTATOR FUNCTION TABLE)
                        4960 ;       INDEX J (INDEX TO THE RBUS DATA TABLE)
                        4961 ;       RTEMP0 (CONTAINS THE DRIVEN RBUS DATA PATTERN)
                        4962 ;       MTEMP1 (CONTAINS THE RECEIVED WBUS DATA PATTERN)
                        4963 ;
                        4964 ;--
                        4965 ;*****************************************************************************
                        4966
                        4967         SUBTEST         1
                        0029
                        0029        ;////////////////////////////////////////////////////////////////////////
                        4968 ;+
                        4969 ; This subtest examines the BUT/SRKSTA function for SRKSTA<1> = DSIZE<1> and
                        4970 ; SRKSTA<0> = DSIZE<0>.
                        4971 ;-
                        4972         INITIALIZE                              ; INITIALZE THE CPU
                        4973         LOOP            I,1,8                   ; LOOP FOR 8 ROT FUNCTIONS
                        4974         LDFIELD         4$,I,B,101$,58,63       ; LOADS ROT FIELD
                        4975         LOOP            J,1,3                   ; LOOP FOR 3 DATA TYPES
                        4976         LDFIELD         1$,J,B,101$,40,41       ; LOADS TEST DTYPE FIELD
                        4977         LOADDCS         100$,,0,3               ; LOAD TEST MICRO-CODE
                        4978         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        4979         FETCH           0                       ; EXECUTE DSIZE SRK BRANCH
                        4980         BRSTCLK         2                       ; ...
                        4981         CMPREGMSK       CSTRP,2$,J,3$,,W        ; IS THE CS ADDRESS CORRECT?
                        4982         IFERROR         ,<SRK>                  ; IF NOT, SIGNAL AN ERROR
                        4983         ENDLOOP         J                       ; LOOP FOR REMAINING DTYPES
                        4984         ENDLOOP         I                       ; LOOP FOR REMAINING FUNCTIONS
                        4985
                        4986         SUBTEST         2
                        0088
                        0088        ;////////////////////////////////////////////////////////////////////////
                        4987 ;+
                        4988 ; This subtest examines the BUT/SRKSTA function for SRKSTA<1> = S<3:0>.NE.0 and
                        4989 ; SRKSTA<0> = S<3:0>.NE.(11,13).
                        4990 ;-
                        4991         LOADDCS         102$,,0,4               ; LOAD THE TEST MICROCODE
                        4992         LOOP            I,1,24                  ; LOOP FOR 24 MBUS PATTERNS
                        4993         LOADREG         WDREG,5$,I,L            ; LOAD WDREG WITH PATTERN
                        4994         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        4995         FETCH           0                       ; EXECUTE THE TEST
                        4996         BRSTCLK         3                       ; ...
                        4997         CMPREGMSK       CSTRP,6$,I,3$,,W        ; IS THE CS ADDRESS CORRECT?
                        4998         IFERROR         2,<SRK>                 ; IF NOT, SIGNAL AN ERROR
                        4999         ENDLOOP         I                       ; CONTINUE FOR THE REST
                        5000
                        5001         SUBTEST         3
                        00C1
                        00C1        ;////////////////////////////////////////////////////////////////////////
                        5002 ;+
                        5003 ; This subtest examines the BUT/SRKSTA function for SRKSTA<1> = WX<31:16>.NE.0
                        5004 ; and SRKSTA<0> = WX<15:00>.NE.0.
                        5005 ;-
```

```
                        00C1  5006          LOOP            I,1,2                   ; LOOP FOR 2 ROT FUNCTIONS
                        00C8  5007          LDFIELD         9$,I,B,104$,58,63       ; LOADS ROT FIELD
                        00D9  5008          LOADDCS         103$,,0,4               ; LOAD TEST MICROCODE
                        00E0  5009          LOOP            J,1,19                  ; LOOP ON J FOR 19 PATTERNS
                        00E7  5010          LOADREG         WDREG,7$,J,L            ; LOAD WDREG WITH TEST PATTERN
                        00EF  5011          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        00F1  5012          FETCH           0                       ; EXECUTE THE TEST
                        00F6  5013          BRSTCLK         3                       ; ...
                        00FA  5014          CMPREGMSK       CSTRP,8$,J,3$,,W        ; IS THE CS ADDRESS CORRECT?
                        0106  5015          IFERROR         2,<SRK>                 ; IF NOT, SIGNAL AN ERROR
                        010C  5016          ENDLOOP         J                       ; LOOP FOR REMAINING PATTERNS
                        010F  5017          ENDLOOP         I                       ; LOOP FOR REMAINING FUNCTIONS
                        0112  5018          SKIP                                    ; END OF TEST
                        0119  5019
                        0119  5020 BEGIN_TEST_DATA
                        0119
                        0119          ;-------------------------------------------------------------------------
                        0119  5021
                        0119  5022 ; Data for Subtest 1
                        0119  5023
                     00 011A  5024 1$:      .BYTE   ^X00    ; DTYPE = BYTE
                     01 011A  5025          .BYTE   ^X01    ; DTYPE = WORD
                     02 011B  5026          .BYTE   ^X02    ; DTYPE = LONG
                        011C  5027
           1802 1801 1800 011C  5028 2$:      .WORD   ^X1800,^X1801,^X1802    ; EXPECTED CS ADDRESSES
                   3FFF 0122  5029 3$:      .WORD   ^X3FFF                  ; ADDRESS COMPARISON MASK
                        0124  5030
                     08 0124  5031 4$:      .BYTE   ^X08    ; ROT/RL.RM.4
                     09 0125  5032          .BYTE   ^X09    ; ROT/RR.MR.4
                     0A 0126  5033          .BYTE   ^X0A    ; ROT/RR.RR.SIZ
                     0B 0127  5034          .BYTE   ^X0B    ; ROT/RR.MR.9
                     0C 0128  5035          .BYTE   ^X0C    ; ROT/XZ.PTX
                     0D 0129  5036          .BYTE   ^X0D    ; ROT/XZ.VPN
                     0E 012A  5037          .BYTE   ^X0E    ; ROT/RR.MM.SIZ
                     0F 012B  5038          .BYTE   ^X0F    ; ROT/GETNIB
                        012C  5039
                        012C  5040 ; Data for Subtest 2
                        012C  5041
   C0CCCCCC 50555555 A0AAAAAA 012C  5042 5$:      .LONG   ^XA0AAAAAA,^X50555555,^XC0CCCCCC ; SRKSTA<1:0> = 01
   0000FFFF 00FF00FF F0F0F0F0 0138  5043          .LONG   ^XF0F0F0F0,^X00FF00FF,^X0000FFFF
   3B333333 5B555555 ABAAAAAA 0144  5044          .LONG   ^XABAAAAAA,^X5B555555,^X3B333333 ; SRKSTA<1:0> = 10
   FBFF0000 FB00FF00 0B0F0F0F 0150  5045          .LONG   ^X0B0F0F0F,^XFB00FF00,^XFBFF0000
   3D333333 5D555555 ADAAAAAA 015C  5046          .LONG   ^XADAAAAAA,^X5D555555,^X3D333333
   FDFF0000 FD00FF00 0D0F0F0F 0168  5047          .LONG   ^X0D0F0F0F,^XFD00FF00,^XFDFF0000
   33333333 55555555 AAAAAAAA 0174  5048          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; SRKSTA<1:0> = 11
   FFFF0000 FF00FF00 0F0F0F0F 0180  5049          .LONG   ^X0F0F0F0F,^XFF00FF00,^XFFFF0000
                        018C  5050
1801 1801 1801 1801 1801 1801 018C  5051 6$:      .WORD   ^X1801,^X1801,^X1801,^X1801,^X1801,^X1801
1802 1802 1802 1802 1802 1802 0198  5052          .WORD   ^X1802,^X1802,^X1802,^X1802,^X1802,^X1802
1802 1802 1802 1802 1802 1802 01A4  5053          .WORD   ^X1802,^X1802,^X1802,^X1802,^X1802,^X1802
1803 1803 1803 1803 1803 1803 01B0  5054          .WORD   ^X1803,^X1803,^X1803,^X1803,^X1803,^X1803
                        01BC  5055
                        01BC  5056 ; Data for Subtest 3
                        01BC  5057
                00000000 01BC  5058 7$:      .LONG   ^X00000000                      ; SRKSTA<1:0> = 00
```

```
        00003333 00005555 0000AAAA  01C0  5059          .LONG   ^X0000AAAA,^X00005555,^X00003333        ; SRKSTA<1:0> = 01
        0000FFFF 000000FF 00000F0F  01CC  5060          .LONG   ^X00000F0F,^X000000FF,^X0000FFFF
        CCCC0000 55550000 AAAA0000  01D8  5061          .LONG   ^XAAAA0000,^X55550000,^XCCCC0000        ; SRKSTA<1:0> = 10
        FFFF0000 FF000000 F0F00000  01E4  5062          .LONG   ^XF0F00000,^XFF000000,^XFFFF0000
        33333333 55555555 AAAAAAAA  01F0  5063          .LONG   ^XAAAAAAAA,^X55555555,^X33333333        ; SRKSTA<1:0> = 11
        FFFFFFFF 00FF00FF 0F0F0F0F  01FC  5064          .LONG   ^X0F0F0F0F,^X00FF00FF,^XFFFFFFFF
                                    0208  5065
                             1800   0208  5066 8$:      .WORD   ^X1800
    1801 1801 1801 1801 1801 1801   020A  5067          .WORD   ^X1801,^X1801,^X1801,^X1801,^X1801,^X1801
    1802 1802 1802 1802 1802 1802   0216  5068          .WORD   ^X1802,^X1802,^X1802,^X1802,^X1802,^X1802
    1803 1803 1803 1803 1803 1803   0222  5069          .WORD   ^X1803,^X1803,^X1803,^X1803,^X1803,^X1803
                                    022E  5070
                               27   022E  5071 9$:      .BYTE   ^X27    ; ROT/WX.NE.0.113.D
                               23   022F  5072          .BYTE   ^X23    ; ROT/WX.NE.0.103
                                    0230  5073
                                    0230  5074 ; Microcode for Subtest 1
                                    0230  5075
                                    0230  5076 100$:
U 00, 7700,C800,0364,0300,0470,1801 0230  5077 ;% 00   NOP
                                    023B  5081 101$:
U 01, 7701,0800,9F64,DCD8,0470,1800 023B  5082 ;% 01   RSRC/ZERO,DTYPE/BYTE,ROT/PL_MSS,BUT/SRKSTA,NEXT/1800,VSIZE/1
U 02, 7300,4800,0364,0300,0470,1803 0246  5086 ;% 02   STOP.CLOCK
                                    0251  5090
                                    0251  5091 ; Microcode for Subtest 2
                                    0251  5092
                                    0251  5093 102$:
U 00, 7700,C800,0364,0300,0470,1801 0251  5094 ;% 00   NOP
U 01, 5700,C860,0364,0300,0470,1802 025C  5098 ;% 01   M[TEMP0]_RDM
U 02, 7700,4840,6347,DF04,0470,1800 0267  5102 ;% 02   MSRC/TEMP0,ROT/BCDSWP,ALPCTL/WX_S.Q_0,RSRC/TEMP1,SPW/RLONG,BUT/SRKSTA,NEXT/1
U 03, 7300,C800,0364,0300,0470,1804 0272  5106 ;% 03   STOP.CLOCK
                                    027D  5110
                                    027D  5111 ; Microcode for Subtest 3
                                    027D  5112
                                    027D  5113 103$:
U 00, 7700,C800,0364,0300,0470,1801 027D  5114 ;% 00   NOP
U 01, 5700,8840,0364,0300,0470,1802 0288  5118 ;% 01   R[TEMP0]_RDM
                                    0293  5122 104$:
U 02, 7700,4861,9E47,DF00,0470,1800 0293  5123 ;% 02   RSRC/TEMP0,ALPCTL/WX_R.Q_D,MSRC/TEMP1,SPW/MLONG,ROT/PL_MSS,BUT/SRKSTA,NEXT/1
U 03, 7300,C800,0364,0300,0470,1804 029E  5127 ;% 03   STOP.CLOCK
                                    02A9  5131
                                    02A9  5132 END_TEST_DATA
                                    02A9
                                    02A9       ;-------------------------------------------------------------------------
                                    02A9  5133
                                    02A9  5134 ENDTEST
```

```
0026        .SBTTL  "TEST   0A3     Rotator Test 52 SRK Status Test 3
0026  5136 ;*******************************************************************************
0026  5137 ;++
0026  5138 ;
0026  5139 ; FUNCTIONAL DESCRIPTION
0026  5140 ;
0026  5141 ;       This test examines the SRK logic that implements the BUT/SRKSTA micro-
0026  5142 ;       branch based on the ASCII SIGN CHECK function. The BUT/SRKSTA micro-
0026  5143 ;       branch is performed by placing two bits called SRKSTA<1:0> onto the
0026  5144 ;       Control Store Address Bus (CSADDR) bits <1:0>.
0026  5145 ;
0026  5146 ; TEST ALGORITHM
0026  5147 ;
0026  5148 ;       The test algorithm is as follows:
0026  5149 ;
0026  5150 ;        1. Initialize the CPU.
0026  5151 ;        2. Start signature analysis.
0026  5152 ;        3. Prepare a loop on I for the 8 rotator microorders that perform the
0026  5153 ;           ASCII SIGN CHECK function on the SRK status lines.
0026  5154 ;        4. Load the ROT field of the test microinstruction for the function to
0026  5155 ;           be tested for this iteration of the I loop.
0026  5156 ;        5. Load the test microinstruction into DCS at address 1801.
0026  5157 ;        6. Prepare to loop on J for the 24 WBUS patterns used to examine the
0026  5158 ;           ASCII SIGN CHECK function.
0026  5159 ;        7. Load a WBUS test pattern from a table indexed by J into the WDREG.
0026  5160 ;        8. Loop on error from here.
0026  5161 ;        9. Execute the following microcode:
0026  5162 ;               1. NOP to get the CPU clock up to speed.
0026  5163 ;               2. Source WDREG onto the WBUS, write it into RTEMP0 and execute
0026  5164 ;                  the rotator and BUT/SRKSTA operations needed to perform the
0026  5165 ;                  ASCII SIGN CHECK function.
0026  5166 ;               3. Stop the CPU clock to halt.
0026  5167 ;       10. Mask out bit 13 of the microaddress in the Control Store Address
0026  5168 ;           Trap Register (CSTRP) and compare it against the expected data from
0026  5169 ;           a table indexed by J.
0026  5170 ;       11. If the two do not match, signal an error.
0026  5171 ;       12. Continue the loop on J for the remaining WBUS patterns.
0026  5172 ;       13. Continue the loop on I for the remaining rotator functions.
0026  5173 ;       14. Stop signature analysis.
0026  5174 ;       15. End of the test, skip to the next test.
0026  5175 ;
0026  5176 ; TEST DATA
0026  5177 ;
0026  5178 ;       The data used in this test consists of 24 WBUS data patterns that are
0026  5179 ;       input to the rotator from the WBUS in order to examine the ASCII SIGN
0026  5180 ;       CHECK SRK status microbranch. The data is as follows:
0026  5181 ;
0026  5182 ;            INDEX I    ROTATOR FUNCTION
0026  5183 ;
0026  5184 ;               01      14 = ROT/CLR3BM
0026  5185 ;               02      15 = ROT/ASL.R.7
0026  5186 ;               03      16 = ROT/ZERO
0026  5187 ;               04      17 = ROT/ASL.R.SIZ
0026  5188 ;               05      1C = ROT/CONX.SIZ
0026  5189 ;               06      1D = ROT/ASR.M.3
```

```
0026  5190 ;                07      1E = ROT/FPLIT
0026  5191 ;                08      1F = ROT/CVTNP
0026  5192 ;
0026  5193 ;             INDEX J      WBUS       CS ADDRESS
0026  5194 ;
0026  5195 ;                01      AAAAAA2D        1800
0026  5196 ;                02      5555552D        1800
0026  5197 ;                03      3333332D        1800
0026  5198 ;                04      0F0F0F2D        1800
0026  5199 ;                05      00FF002D        1800
0026  5200 ;                06      0000FF2D        1800
0026  5201 ;                07      AAAAAA20        1801
0026  5202 ;                08      55555520        1801
0026  5203 ;                09      33333320        1801
0026  5204 ;                0A      0F0F0F20        1801
0026  5205 ;                0B      00FF0020        1801
0026  5206 ;                0C      0000FF20        1801
0026  5207 ;                0D      AAAAAA2B        1801
0026  5208 ;                0E      5555552B        1801
0026  5209 ;                0F      3333332B        1801
0026  5210 ;                10      0F0F0F2B        1801
0026  5211 ;                11      00FF002B        1801
0026  5212 ;                12      0000FF2B        1801
0026  5213 ;                13      AAAAAAAA        1803
0026  5214 ;                14      55555555        1803
0026  5215 ;                15      33333333        1803
0026  5216 ;                16      0F0F0F0F        1803
0026  5217 ;                17      00FF00FF        1803
0026  5218 ;                18      0000FFFF        1803
0026  5219 ;
0026  5220 ; LOGIC DESCRIPTION
0026  5221 ;
0026  5222 ;       A combination of discrete logic on the DPM module and logic internal to
0026  5223 ;       the SRK gate array is used to implement the SRK status microbranch
0026  5224 ;       ASCII SIGN CHECK. The SRK gate array receives the ROT field of the
0026  5225 ;       microword and WBUS<7:0>. This enables it to generate the signals SRK
0026  5226 ;       ST1 and SRK ST0 based on the contents of the ROT field and WBUS<7:0>.
0026  5227 ;       These signals are passed to two 74S151 8 to 1 multiplexers that can
0026  5228 ;       drive bits 0 and 1 of the CS ADDRESS BUS. The mux that receives SRK ST1
0026  5229 ;       will select this signal and enable its output whenever it receives a
0026  5230 ;       BUT/SRKSTA microbranch. The mux that receives SRK ST0 will select this
0026  5231 ;       signal and enable its output whenever it receives a BUT/SRKSTA or
0026  5232 ;       BUT/CCBR0.SRKSTA0 microbranch. For this test only BUT/SRKSTA is used.
0026  5233 ;
0026  5234 ; ASSUMPTIONS
0026  5235 ;
0026  5236 ;       This test assumes the following DPM logic has been tested and is
0026  5237 ;       working; the RBUS and WBUS; and the dual port scratchpad register
0026  5238 ;       RTEMP0.
0026  5239 ;
0026  5240 ; ERROR DESCRIPTION
0026  5241 ;
0026  5242 ;       On error this tell will call out the SRK gate array and will type out
0026  5243 ;       the following information:
0026  5244 ;
```

```
                                0026  5245 ;       EXPECTED CONTROL STORE ADDRESS
                                0026  5246 ;       RECEIVED CONTROL STORE ADDRESS
                                0026  5247 ;       INDEX I (INDEX TO THE ROTATOR FUNCTION UNDER TEST)
                                0026  5248 ;       INDEX J (INDEX TO THE WBUS DATA TABLE)
                                0026  5249 ;       RTEMP0 (WBUS CONTENTS)
                                0026  5250 ;
                                0026  5251 ;--
                                0026  5252 ;*******************************************************************************
                                0026  5253 ;///////////////////////////////////////////////////////////////////////////////
                                0026  5254
                                0026  5255         INITIALIZE                                      ; INITIALIZE THE CPU
                                0028  5256         LOOP            I,1,8                           ; LOOP FOR 8 ASCII SIGN OPCODES
                                002F  5257         LDFIELD         4$,I,B,10$,58,63                ; LOAD THE ROT FUNCTION TO TEST
                                0040  5258         LOADDCS         10$,,1,1                        ; LOAD TEST INST. INTO DCS
                                0047  5259         LOOP            J,1,24                          ; LOOP FOR 24 WBUS PATTERNS
                                004E  5260         LOADREG         WDREG,1$,J,L                    ; LOAD WDREG WITH TEST PATTERN
                                0056  5261         ERRLOOP                                         ; LOOP ON ERROR FROM HERE
                                0058  5262         FETCH           0                               ; EXECUTE THE TEST
                                005D  5263         BRSTCLK         2                               ; ...
                                0061  5264         CMPREGMSK       CSTRP,2$,J,3$,,W                ; IS THE CS ADDRESS CORRECT?
                                006D  5265         IFERROR         1,<SRK>                         ; IF NOT, SIGNAL AN ERROR
                                0073  5266         ENDLOOP         J                               ; CONTINUE FOR OTHER PATTERNS
                                0076  5267         ENDLOOP         I                               ; CONTINUE FOR OTHER MICROORDS
                                0079  5268         SKIP                                            ; END OF TEST
                                0080  5269
                                0080  5270 BEGIN_TEST_DATA
                                0080
                                0080         ;-------------------------------------------------------------------------
                                0080  5271
                                0080  5272 ; WBUS Data Table
                                0080  5273
 3333332D 5555552D AAAAAA2D     0080  5274 1$:      .LONG   ^XAAAAAA2D,^X5555552D,^X3333332D ; SRKSTA<1:0> = 00
 0000FF2D 00FF002D 0F0F0F2D     008C  5275          .LONG   ^X0F0F0F2D,^X00FF002D,^X0000FF2D
 33333320 55555520 AAAAAA20     0098  5276          .LONG   ^XAAAAAA20,^X55555520,^X33333320 ; SRKSTA<1:0> = 01
 0000FF20 00FF0020 0F0F0F20     00A4  5277          .LONG   ^X0F0F0F20,^X00FF0020,^X0000FF20
 3333332B 5555552B AAAAAA2B     00B0  5278          .LONG   ^XAAAAAA2B,^X5555552B,^X3333332B
 0000FF2B 00FF002B 0F0F0F2B     00BC  5279          .LONG   ^X0F0F0F2B,^X00FF002B,^X0000FF2B
 33333333 55555555 AAAAAAAA     00C8  5280          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; SRKSTA<1:0> = 11
 0000FFFF 00FF00FF 0F0F0F0F     00D4  5281          .LONG   ^X0F0F0F0F,^X00FF00FF,^X0000FFFF
                                00E0  5282
                                00E0  5283 ; Expected Control Store Address Table
                                00E0  5284
 1800 1800 1800 1800 1800 1800  00E0  5285 2$:      .WORD   ^X1800,^X1800,^X1800,^X1800,^X1800,^X1800
 1801 1801 1801 1801 1801 1801  00EC  5286          .WORD   ^X1801,^X1801,^X1801,^X1801,^X1801,^X1801
 1801 1801 1801 1801 1801 1801  00F8  5287          .WORD   ^X1801,^X1801,^X1801,^X1801,^X1801,^X1801
 1803 1803 1803 1803 1803 1803  0104  5288          .WORD   ^X1803,^X1803,^X1803,^X1803,^X1803,^X1803
                                0110  5289
                                0110  5290 ; Control Store Address Comparison Mask
                                0110  5291
                          3FFF  0110  5292 3$:      .WORD   ^X3FFF
                                0112  5293
                                0112  5294 ; Rotator Test Functions
                                0112  5295
                            14  0112  5296 4$:      .BYTE   ^X14    ; ROT/CLR3BM
                            15  0113  5297          .BYTE   ^X15    ; ROT/ASL.R.7
```

```
                      16  0114  5298          .BYTE   ^X16     ; ROT/ZERO
                      17  0115  5299          .BYTE   ^X17     ; ROT/ASL.R.SIZ
                      1C  0116  5300          .BYTE   ^X1C     ; ROT/CONX.SIZ
                      1D  0117  5301          .BYTE   ^X1D     ; ROT/ASR.M.3
                      1E  0118  5302          .BYTE   ^X1E     ; ROT/FPLIT
                      1F  0119  5303          .BYTE   ^X1F     ; ROT/CVTNP
                          011A  5304
                          011A  5305 ; Reloadable Test Microword
                          011A  5306
                          011A  5307 10$:
U 01, 5700,4840,5364,DF00,0470,1800   011A  5308 ;% 01   WB_RDM,RSRC/TEMP0,SPW/RLONG,ROT/CLR3BM,BUT/SRKSTA,NEXT/1800
                          0125  5312
                          0125  5313 BEGIN_CPU_DATA
                          0002  5314
                          0002  5315 DCS_DATA
                          0001  5316
U 00, 7700,C800,0364,0300,0470,1801   0001  5317 ;% 00   NOP
U 01, 5700,4840,5364,DF00,0470,1800   000C  5321 ;% 01   WB_RDM,RSRC/TEMP0,SPW/RLONG,ROT/CLR3BM,BUT/SRKSTA,NEXT/1800
U 02, 7300,4800,0364,0300,0470,1803   0017  5325 ;% 02   STOP.CLOCK
                          0022  5329
                          0022  5330 RTEMP_DATA
                          0001  5331
                 00000000 0001  5332          .LONG   0       ; INITIALIZES RTEMP0
                          0005  5333
                          0005  5334 END_CPU_DATA
                          0125  5335
                          0125  5336 END_TEST_DATA
                          0125
                          0125       ;-----------------------------------------------------------------------------
                          0125  5337
                          0125  5338 ENDTEST
```

```
                        0026        .SBTTL  "TEST   0A4     Rotator Test 53 SRK Status Test 4
                        0026  5340 ;*******************************************************************************
                        0026  5341 ;++
                        0026  5342 ;
                        0026  5343 ; FUNCTIONAL DESCRIPTION
                        0026  5344 ;
                        0026  5345 ;       This test examines the SRK logic that implements the BUT/SRKSTA micro-
                        0026  5346 ;       branch based on the WBUS RANGE CHECK function. The BUT/SRKSTA micro-
                        0026  5347 ;       branch is performed by placing two bits called SRKSTA<1:0> onto the
                        0026  5348 ;       Control Store Address Bus (CSADDR) bits <1:0>.
                        0026  5349 ;
                        0026  5350 ; TEST ALGORITHM
                        0026  5351 ;
                        0026  5352 ;       The test algorithm is as follows:
                        0026  5353 ;
                        0026  5354 ;        1. Initialize the CPU.
                        0026  5355 ;        2. Start signature analysis.
                        0026  5356 ;        3. Prepare a loop on I for the 4 rotator microorders that perform the
                        0026  5357 ;           WBUS RANGE CHECK function on the SRK status lines.
                        0026  5358 ;        4. Load the ROT field of the test microinstruction for the function to
                        0026  5359 ;           be tested for this iteration of the I loop.
                        0026  5360 ;        5. Load the test microinstruction into DCS at address 1801.
                        0026  5361 ;        6. Prepare to loop on J for the 30 WBUS patterns used to examine the
                        0026  5362 ;           WBUS RANGE CHECK function.
                        0026  5363 ;        7. Load a WBUS test pattern from a table indexed by J into the WDREG.
                        0026  5364 ;        8. Loop on error from here.
                        0026  5365 ;        9. Execute the following microcode:
                        0026  5366 ;                1. NOP to get the CPU clock up to speed.
                        0026  5367 ;                2. Source WDREG onto the WBUS, write it into RTEMP0 and execute
                        0026  5368 ;                   the rotator and BUT/SRKSTA operations needed to perform the
                        0026  5369 ;                   WBUS RANGE CHECK function.
                        0026  5370 ;                3. Stop the CPU clock to halt.
                        0026  5371 ;       10. Mask out bit 13 of the microaddress in the Control Store Address
                        0026  5372 ;           Trap Register (CSTRP) and compare it against the expected data from
                        0026  5373 ;           a table indexed by J.
                        0026  5374 ;       11. If the two do not match, signal an error.
                        0026  5375 ;       12. Continue the loop on J for the remaining WBUS patterns.
                        0026  5376 ;       13. Continue the loop on I for the remaining rotator functions.
                        0026  5377 ;       14. Stop signature analysis.
                        0026  5378 ;       15. End of the test, skip to the next test.
                        0026  5379 ;
                        0026  5380 ; TEST DATA
                        0026  5381 ;
                        0026  5382 ;       The data used in this test consists of 24 WBUS data patterns that are
                        0026  5383 ;       input to the rotator from the WBUS in order to examine the WBUS RANGE
                        0026  5384 ;       CHECK SRK status microbranch. The data is as follows:
                        0026  5385 ;
                        0026  5386 ;            INDEX I    ROTATOR FUNCTION
                        0026  5387 ;
                        0026  5388 ;               01      2D = ROT/PL.SL_WB
                        0026  5389 ;               02      2F = ROT/SL.PL_WB
                        0026  5390 ;               03      2C = ROT/PL
                        0026  5391 ;               04      2E = ROT/SL
                        0026  5392 ;
                        0026  5393 ;            INDEX J      WBUS       CS ADDRESS
```

```
0026  5394 ;
0026  5395 ;                01      AAAAAA01        1800
0026  5396 ;                02      55555501        1800
0026  5397 ;                03      33333301        1800
0026  5398 ;                04      0F0F0F01        1800
0026  5399 ;                05      00FF0001        1800
0026  5400 ;                06      0000FF01        1800
0026  5401 ;                07      AAAAAA1F        1800
0026  5402 ;                08      5555551F        1800
0026  5403 ;                09      3333331F        1800
0026  5404 ;                0A      0F0F0F1F        1800
0026  5405 ;                0B      00FF001F        1800
0026  5406 ;                0C      0000FF1F        1800
0026  5407 ;                0D      AAAAAA00        1801
0026  5408 ;                0E      55555500        1801
0026  5409 ;                0F      33333300        1801
0026  5410 ;                10      0F0F0F00        1801
0026  5411 ;                11      00FF0000        1801
0026  5412 ;                12      0000FF00        1801
0026  5413 ;                13      AAAAAA20        1802
0026  5414 ;                14      55555520        1802
0026  5415 ;                15      33333320        1802
0026  5416 ;                16      0F0F0F20        1802
0026  5417 ;                17      00FF0020        1802
0026  5418 ;                18      0000FF20        1802
0026  5419 ;                19      AAAAAAAA        1803
0026  5420 ;                1A      55555555        1803
0026  5421 ;                1B      33333333        1803
0026  5422 ;                1C      0F0F0F2F        1803
0026  5423 ;                1D      00FF00FF        1803
0026  5424 ;                1E      0000FFFF        1803
0026  5425 ;
0026  5426 ; LOGIC DESCRIPTION
0026  5427 ;
0026  5428 ;       A combination of discrete logic on the DPM module and logic internal to
0026  5429 ;       the SRK gate array is used to implement the SRK status microbranch WBUS
0026  5430 ;       RANGE CHECK. The SRK gate array receives the ROT field of the microword
0026  5431 ;       and WBUS bits <7:0>. This enables it to generate the signals SRK ST1
0026  5432 ;       and SRK ST0 based on the contents of the ROT field and WBUS<7:0>. These
0026  5433 ;       signals are passed to two 74S151 8 to 1 multiplexers that can drive
0026  5434 ;       bits 0 and 1 of the CS ADDRESS BUS. The mux that receives SRK ST1 will
0026  5435 ;       select this signal and enable its output whenever it receives a
0026  5436 ;       BUT/SRKSTA microbranch. The mux that receives SRK ST0 will select this
0026  5437 ;       signal and enable its output whenever it receives a BUT/SRKSTA or
0026  5438 ;       BUT/CCBR0.SRKSTA0 microbranch. For this test only BUT/SRKSTA is used.
0026  5439 ;
0026  5440 ; ASSUMPTIONS
0026  5441 ;
0026  5442 ;       This test assumes the following DPM logic has been tested and is
0026  5443 ;       working; the RBUS and WBUS; and the dual port scratchpad register
0026  5444 ;       RTEMP0.
0026  5445 ;
0026  5446 ; ERROR DESCRIPTION
0026  5447 ;
0026  5448 ;       On error this tell will call out the SRK gate array and will type out
```

```
                                0026  5449 ;       the following information:
                                0026  5450 ;
                                0026  5451 ;       EXPECTED CONTROL STORE ADDRESS
                                0026  5452 ;       RECEIVED CONTROL STORE ADDRESS
                                0026  5453 ;       INDEX I (INDEX TO THE ROTATOR FUNCTION TABLE)
                                0026  5454 ;       INDEX J (INDEX TO THE WBUS AND CS ADDRESS DATA TABLES)
                                0026  5455 ;       RTEMP0 (WBUS CONTENTS)
                                0026  5456 ;
                                0026  5457 ;--
                                0026  5458 ;*******************************************************************************
                                0026  5459 ;///////////////////////////////////////////////////////////////////////////////
                                0026  5460
                                0026  5461         INITIALIZE                                      ; INITIALIZE THE CPU
                                0028  5462         LOOP            I,1,4                           ; LOOP FOR 4 WB RANGE OPCODES
                                002F  5463         LDFIELD         4$,I,B,10$,58,63                ; LOADS ROT FIELD
                                0040  5464         LOADDCS         10$,,1,1                        ; LOAD TEST INST. INTO DCS
                                0047  5465         LOOP            J,1,30                          ; LOOP FOR 30 WBUS PATTERNS
                                004E  5466         LOADREG         WDREG,1$,J,L                    ; LOAD WDREG WITH TEST PATTERN
                                0056  5467         ERRLOOP                                         ; LOOP ON ERROR FROM HERE
                                0058  5468         FETCH           0                               ; EXECUTE THE TEST
                                005D  5469         BRSTCLK         2                               ; ...
                                0061  5470         CMPREGMSK       CSTRP,2$,J,3$,,W                ; IS THE CS ADDRESS CORRECT?
                                006D  5471         IFERROR         1,<SRK>                         ; IF NOT, SIGNAL AN ERROR
                                0073  5472         ENDLOOP         J                               ; CONTINUE FOR OTHER PATTERNS
                                0076  5473         ENDLOOP         I                               ; CONTINUE FOR OTHER MICROORDS
                                0079  5474         SKIP                                            ; END OF TEST
                                0080  5475
                                0080  5476 BEGIN_TEST_DATA
                                0080
                                0080        ;-------------------------------------------------------------------------
                                0080  5477
                                0080  5478 ; WBUS Data Table
                                0080  5479
   33333301 55555501 AAAAAA01   0080  5480 1$:      .LONG   ^XAAAAAA01,^X55555501,^X33333301 ; SRKSTA<1:0> = 00
   0000FF01 00FF0001 0F0F0F01   008C  5481          .LONG   ^X0F0F0F01,^X00FF0001,^X0000FF01
   3333331F 5555551F AAAAAA1F   0098  5482          .LONG   ^XAAAAAA1F,^X5555551F,^X3333331F
   0000FF1F 00FF001F 0F0F0F1F   00A4  5483          .LONG   ^X0F0F0F1F,^X00FF001F,^X0000FF1F
   33333300 55555500 AAAAAA00   00B0  5484          .LONG   ^XAAAAAA00,^X55555500,^X33333300 ; SRKSTA<1:0> = 01
   0000FF00 00FF0000 0F0F0F00   00BC  5485          .LONG   ^X0F0F0F00,^X00FF0000,^X0000FF00
   33333320 55555520 AAAAAA20   00C8  5486          .LONG   ^XAAAAAA20,^X55555520,^X33333320 ; SRKSTA<1:0> = 10
   0000FF20 00FF0020 0F0F0F20   00D4  5487          .LONG   ^X0F0F0F20,^X00FF0020,^X0000FF20
   33333333 55555555 AAAAAAAA   00E0  5488          .LONG   ^XAAAAAAAA,^X55555555,^X33333333 ; SRKSTA<1:0> = 11
   0000FFFF 00FF00FF 0F0F0F2F   00EC  5489          .LONG   ^X0F0F0F2F,^X00FF00FF,^X0000FFFF
                                00F8  5490
                                00F8  5491 ; Expected Control Store Address Table
                                00F8  5492
1800 1800 1800 1800 1800 1800   00F8  5493 2$:      .WORD   ^X1800,^X1800,^X1800,^X1800,^X1800,^X1800
1800 1800 1800 1800 1800 1800   0104  5494          .WORD   ^X1800,^X1800,^X1800,^X1800,^X1800,^X1800
1801 1801 1801 1801 1801 1801   0110  5495          .WORD   ^X1801,^X1801,^X1801,^X1801,^X1801,^X1801
1802 1802 1802 1802 1802 1802   011C  5496          .WORD   ^X1802,^X1802,^X1802,^X1802,^X1802,^X1802
1803 1803 1803 1803 1803 1803   0128  5497          .WORD   ^X1803,^X1803,^X1803,^X1803,^X1803,^X1803
                                0134  5498
                                0134  5499 ; Control Store Address Comparison Mask
                                0134  5500
                        3FFF    0134  5501 3$:      .WORD   ^X3FFF
```

```
                                0136  5502
                                0136  5503 ; ROT Field Microorders
                                0136  5504
                             2D 0136  5505 4$:     .BYTE   ^X2D    ; ROT/PL.SL_WB
                             2F 0137  5506         .BYTE   ^X2F    ; ROT/SL.PL_WB
                             2C 0138  5507         .BYTE   ^X2C    ; ROT/PL
                             2E 0139  5508         .BYTE   ^X2E    ; ROT/SL
                                013A  5509
                                013A  5510 ; Reloadable Test Microinstruction
                                013A  5511
                                013A  5512 10$:
U 01, 5700,4840,B764,DF00,0470,1800 013A  5513 ;% 01   WB_RDM,RSRC/TEMP0,SPW/RLONG,ROT/PL.SL_WB,BUT/SRKSTA,NEXT/1800
                                0145  5517
                                0145  5518 BEGIN_CPU_DATA
                                0002  5519
                                0002  5520 DCS_DATA
                                0001  5521
U 00, 7700,C800,0364,0300,0470,1801 0001  5522 ;% 00   NOP
U 01, 5700,4840,B764,DF00,0470,1800 000C  5526 ;% 01   WB_RDM,RSRC/TEMP0,SPW/RLONG,ROT/PL.SL_WB,BUT/SRKSTA,NEXT/1800
U 02, 7300,4800,0364,0300,0470,1803 0017  5530 ;% 02   STOP.CLOCK
                                0022  5534
                                0022  5535 RTEMP_DATA
                                0001  5536
                          00000000 0001  5537         .LONG   0       ; INITIALIZES RTEMP0
                                0005  5538
                                0005  5539 END_CPU_DATA
                                0145  5540
                                0145  5541 END_TEST_DATA
                                0145
                                0145        ;-----------------------------------------------------------------------------
                                0145  5542
                                0145  5543 ENDTEST
```

```
0026          .SBTTL  "TEST   0A5     Rotator Test 54 SRK Status Test 5
0026  5545 ;*******************************************************************************
0026  5546 ;++
0026  5547 ;
0026  5548 ; FUNCTIONAL DESCRIPTION
0026  5549 ;
0026  5550 ;       This test examines the SRK logic that implements the BUT/SRKSTA micro-
0026  5551 ;       branch based on the ABSOLUTE VALUE CHECK function. The BUT/SRKSTA
0026  5552 ;       microbranch is performed by placing two bits called SRKSTA<1:0> onto
0026  5553 ;       the Control Store Address Bus (CSADDR) bits <1:0>.
0026  5554 ;
0026  5555 ; TEST ALGORITHM
0026  5556 ;
0026  5557 ;       The test algorithm is as follows:
0026  5558 ;
0026  5559 ;        1. Initialize the CPU.
0026  5560 ;        2. Start signature analysis.
0026  5561 ;        3. Prepare a loop on I for the 16 rotator microorders that perform the
0026  5562 ;           ABSOLUTE VALUE CHECK function on the SRK status lines.
0026  5563 ;        4. Load the ROT field of the test microinstruction for the function to
0026  5564 ;           be tested for this iteration of the I loop.
0026  5565 ;        5. Load the test microinstruction into DCS at address 1801.
0026  5566 ;        6. Prepare to loop on J for the 54 WBUS patterns used to examine the
0026  5567 ;           ABSOLUTE VALUE CHECK function.
0026  5568 ;        7. Load a WBUS test pattern from a table indexed by J into the WDREG.
0026  5569 ;        8. Loop on error from here.
0026  5570 ;        9. Execute the following microcode:
0026  5571 ;               1. NOP to get the CPU clock up to speed.
0026  5572 ;               2. Source WDREG onto the WBUS, write it into RTEMP0 and execute
0026  5573 ;                  the rotator and BUT/SRKSTA operations needed to perform the
0026  5574 ;                  ABSOLUTE VALUE CHECK function.
0026  5575 ;               3. Stop the CPU clock to halt.
0026  5576 ;       10. Mask out bit 13 of the microaddress in the Control Store Address
0026  5577 ;           Trap Register (CSTRP) and compare it against the expected data from
0026  5578 ;           a table indexed by J.
0026  5579 ;       11. If the two do not match, signal an error.
0026  5580 ;       12. Continue the loop on J for the remaining WBUS patterns.
0026  5581 ;       13. Continue the loop on I for the remaining rotator functions.
0026  5582 ;       14. Stop signature analysis.
0026  5583 ;       15. End of the test, skip to the next test.
0026  5584 ;
0026  5585 ; TEST DATA
0026  5586 ;
0026  5587 ;       The data used in this test consists of 24 WBUS data patterns that are
0026  5588 ;       input to the rotator from the WBUS in order to examine the ABSOLUTE
0026  5589 ;       VALUE CHECK SRK status microbranch. The data is as follows:
0026  5590 ;
0026  5591 ;            INDEX I    ROTATOR FUNCTION
0026  5592 ;
0026  5593 ;               01      3B = ROT/OLIT0.PL_LIT
0026  5594 ;               02      3D = ROT/OLIT0.SL_LIT
0026  5595 ;               03      3F = ROT/OLIT0.PL43_WB
0026  5596 ;               04      30 = ROT/ZLIT0
0026  5597 ;               05      31 = ROT/ZLIT28
0026  5598 ;               06      32 = ROT/ZLIT24
```

```
0026  5599 ;                07      33 = ROT/ZLIT20
0026  5600 ;                08      34 = ROT/ZLIT16
0026  5601 ;                09      35 = ROT/ZLIT12
0026  5602 ;                0A      36 = ROT/ZLIT8
0026  5603 ;                0B      37 = ROT/ZLIT4
0026  5604 ;                0C      38 = ROT/OLIT0
0026  5605 ;                0D      39 = ROT/MINUS1
0026  5606 ;                0E      3A = ROT/OLIT24
0026  5607 ;                0F      3C = ROT/OLIT16
0026  5608 ;                10      3E = ROT/OLIT8
0026  5609 ;
0026  5610 ;              INDEX J     WBUS        CS ADDRESS
0026  5611 ;
0026  5612 ;                01      AAAAAAFF        1800
0026  5613 ;                02      555555FF        1800
0026  5614 ;                03      333333FF        1800
0026  5615 ;                04      0F0F0FFF        1800
0026  5616 ;                05      00FF00FF        1800
0026  5617 ;                06      0000FFFF        1800
0026  5618 ;                07      AAAAAAE1        1800
0026  5619 ;                08      555555E1        1800
0026  5620 ;                09      333333E1        1800
0026  5621 ;                0A      0F0F0FE1        1800
0026  5622 ;                0B      00FF00E1        1800
0026  5623 ;                0C      0000FFE1        1800
0026  5624 ;                0D      AAAAAAE0        1801
0026  5625 ;                0E      555555E0        1801
0026  5626 ;                0F      333333E0        1801
0026  5627 ;                10      0F0F0FE0        1801
0026  5628 ;                11      00FF00E0        1801
0026  5629 ;                12      0000FFE0        1801
0026  5630 ;                13      AAAAAAC0        1801
0026  5631 ;                14      555555C0        1801
0026  5632 ;                15      333333C0        1801
0026  5633 ;                16      0F0F0FC0        1801
0026  5634 ;                17      00FF00C0        1801
0026  5635 ;                18      0000FFC0        1801
0026  5636 ;                19      AAAAAA80        1801
0026  5637 ;                1A      55555580        1801
0026  5638 ;                1B      33333380        1801
0026  5639 ;                1C      0F0F0F80        1801
0026  5640 ;                1D      00FF0080        1801
0026  5641 ;                1E      0000FF80        1801
0026  5642 ;                1F      AAAAAA00        1802
0026  5643 ;                20      55555500        1802
0026  5644 ;                21      33333300        1802
0026  5645 ;                22      0F0F0F00        1802
0026  5646 ;                23      00FF0000        1802
0026  5647 ;                24      0000FF00        1802
0026  5648 ;                25      AAAAAA1F        1802
0026  5649 ;                26      5555551F        1802
0026  5650 ;                27      3333331F        1802
0026  5651 ;                28      0F0F0F1F        1802
0026  5652 ;                29      00FF001F        1802
0026  5653 ;                2A      0000FF1F        1802
```

```
0026  5654 ;                    2B      AAAAAA20        1803
0026  5655 ;                    2C      55555520        1803
0026  5656 ;                    2D      33333320        1803
0026  5657 ;                    2E      0F0F0F20        1803
0026  5658 ;                    2F      00FF0020        1803
0026  5659 ;                    30      0000FF20        1803
0026  5660 ;                    31      AAAAAA40        1803
0026  5661 ;                    32      55555540        1803
0026  5662 ;                    33      33333340        1803
0026  5663 ;                    34      0F0F0F40        1803
0026  5664 ;                    35      00FF0040        1803
0026  5665 ;                    36      0000FF40        1803
0026  5666 ;
0026  5667 ; LOGIC DESCRIPTION
0026  5668 ;
0026  5669 ;       A combination of discrete logic on the DPM module and logic internal to
0026  5670 ;       the SRK gate array is used to implement the SRK status microbranch
0026  5671 ;       ABSOLUTE VALUE CHECK. The SRK gate array receives the ROT field of the
0026  5672 ;       microword and WBUS bits <7:0>. This enables it to generate the signals
0026  5673 ;       SRK ST1 and SRK ST0 based on the contents of the ROT field and
0026  5674 ;       WBUS<7:0>. These signals are passed to two 74S151 8 to 1 multiplexers
0026  5675 ;       that can drive bits 0 and 1 of the CS ADDRESS BUS. The mux that
0026  5676 ;       receives SRK ST1 will select this signal and enable its output whenever
0026  5677 ;       it receives a BUT/SRKSTA microbranch. The mux that receives SRK ST0
0026  5678 ;       will select this signal and enable its output whenever it receives a
0026  5679 ;       BUT/SRKSTA or BUT/CCBR0.SRKSTA0 microbranch. For this test only
0026  5680 ;       BUT/SRKSTA is used.
0026  5681 ;
0026  5682 ; ASSUMPTIONS
0026  5683 ;
0026  5684 ;       This test assumes the following DPM logic has been tested and is
0026  5685 ;       working; the RBUS, MBUS and WBUS; and the dual port scratchpad register
0026  5686 ;       RTEMP0.
0026  5687 ;
0026  5688 ; ERROR DESCRIPTION
0026  5689 ;
0026  5690 ;       On error this tell will call out the SRK gate array and will type out
0026  5691 ;       the following information:
0026  5692 ;
0026  5693 ;       EXPECTED CONTROL STORE ADDRESS
0026  5694 ;       RECEIVED CONTROL STORE ADDRESS
0026  5695 ;       INDEX I (INDEX TO THE ROTATOR FUNCTION UNDER TEST)
0026  5696 ;       INDEX J (INDEX TO THE WBUS AND CS ADDRESS DATA TABLES)
0026  5697 ;       RTEMP0 (WBUS CONTENTS)
0026  5698 ;
0026  5699 ;--
0026  5700 ;*******************************************************************************
0026  5701 ;///////////////////////////////////////////////////////////////////////////////
0026  5702
0026  5703         INITIALIZE                                      ; INITIALIZE THE CPU
0028  5704         LOOP            I,1,16                          ; LOOP FOR 16 ABS VAL OPCODES
002F  5705         LDFIELD         4$,I,B,10$,58,63                ; LOADS ROT FIELD
0040  5706         LOADDCS         10$,,1,1                        ; LOAD TEST INST. INTO DCS
0047  5707         LOOP            J,1,54                          ; LOOP FOR 54 WBUS PATTERNS
004E  5708         LOADREG         WDREG,1$,J,L                    ; LOAD WDREG WITH TEST PATTERN
```

```
                                0056  5709          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                                0058  5710          FETCH           0                       ; EXECUTE THE TEST
                                005D  5711          BRSTCLK         2                       ; ...
                                0061  5712          CMPREGMSK       CSTRP,2$,J,3$,,W        ; IS THE CS ADDRESS CORRECT?
                                006D  5713          IFERROR         1,<SRK>                 ; IF NOT, SIGNAL AN ERROR
                                0073  5714          ENDLOOP         J                       ; CONTINUE FOR OTHER PATTERNS
                                0076  5715          ENDLOOP         I                       ; CONTINUE FOR OTHER MICROORDS
                                0079  5716          SKIP                                    ; END OF TEST
                                0080  5717
                                0080  5718 BEGIN_TEST_DATA
                                0080
                                0080        ;-----------------------------------------------------------------------
                                0080  5719
                                0080  5720 ; WBUS Data Table
                                0080  5721
    333333FF 555555FF AAAAAAFF  0080  5722 1$:      .LONG   ^XAAAAAAFF,^X555555FF,^X333333FF ; SRKSTA<1:0> = 00
    0000FFFF 00FF00FF 0F0F0FFF  008C  5723          .LONG   ^X0F0F0FFF,^X00FF00FF,^X0000FFFF
    333333E1 555555E1 AAAAAAE1  0098  5724          .LONG   ^XAAAAAAE1,^X555555E1,^X333333E1
    0000FFE1 00FF00E1 0F0F0FE1  00A4  5725          .LONG   ^X0F0F0FE1,^X00FF00E1,^X0000FFE1
    333333E0 555555E0 AAAAAAE0  00B0  5726          .LONG   ^XAAAAAAE0,^X555555E0,^X333333E0 ; SRKSTA<1:0> = 01
    0000FFE0 00FF00E0 0F0F0FE0  00BC  5727          .LONG   ^X0F0F0FE0,^X00FF00E0,^X0000FFE0
    333333C0 555555C0 AAAAAAC0  00C8  5728          .LONG   ^XAAAAAAC0,^X555555C0,^X333333C0
    0000FFC0 00FF00C0 0F0F0FC0  00D4  5729          .LONG   ^X0F0F0FC0,^X00FF00C0,^X0000FFC0
    33333380 55555580 AAAAAA80  00E0  5730          .LONG   ^XAAAAAA80,^X55555580,^X33333380
    0000FF80 00FF0080 0F0F0F80  00EC  5731          .LONG   ^X0F0F0F80,^X00FF0080,^X0000FF80
    33333300 55555500 AAAAAA00  00F8  5732          .LONG   ^XAAAAAA00,^X55555500,^X33333300 ; SRKSTA<1:0> = 10
    0000FF00 00FF0000 0F0F0F00  0104  5733          .LONG   ^X0F0F0F00,^X00FF0000,^X0000FF00
    3333331F 5555551F AAAAAA1F  0110  5734          .LONG   ^XAAAAAA1F,^X5555551F,^X3333331F
    0000FF1F 00FF001F 0F0F0F1F  011C  5735          .LONG   ^X0F0F0F1F,^X00FF001F,^X0000FF1F
    33333320 55555520 AAAAAA20  0128  5736          .LONG   ^XAAAAAA20,^X55555520,^X33333320 ; SRKSTA<1:0> = 11
    0000FF20 00FF0020 0F0F0F20  0134  5737          .LONG   ^X0F0F0F20,^X00FF0020,^X0000FF20
    33333340 55555540 AAAAAA40  0140  5738          .LONG   ^XAAAAAA40,^X55555540,^X33333340
    0000FF40 00FF0040 0F0F0F40  014C  5739          .LONG   ^X0F0F0F40,^X00FF0040,^X0000FF40
                                0158  5740
                                0158  5741 ; Expected Control Store Address Table
                                0158  5742
1800 1800 1800 1800 1800 1800   0158  5743 2$:      .WORD   ^X1800,^X1800,^X1800,^X1800,^X1800,^X1800
1800 1800 1800 1800 1800 1800   0164  5744          .WORD   ^X1800,^X1800,^X1800,^X1800,^X1800,^X1800
1801 1801 1801 1801 1801 1801   0170  5745          .WORD   ^X1801,^X1801,^X1801,^X1801,^X1801,^X1801
1801 1801 1801 1801 1801 1801   017C  5746          .WORD   ^X1801,^X1801,^X1801,^X1801,^X1801,^X1801
1801 1801 1801 1801 1801 1801   0188  5747          .WORD   ^X1801,^X1801,^X1801,^X1801,^X1801,^X1801
1802 1802 1802 1802 1802 1802   0194  5748          .WORD   ^X1802,^X1802,^X1802,^X1802,^X1802,^X1802
1802 1802 1802 1802 1802 1802   01A0  5749          .WORD   ^X1802,^X1802,^X1802,^X1802,^X1802,^X1802
1803 1803 1803 1803 1803 1803   01AC  5750          .WORD   ^X1803,^X1803,^X1803,^X1803,^X1803,^X1803
1803 1803 1803 1803 1803 1803   01B8  5751          .WORD   ^X1803,^X1803,^X1803,^X1803,^X1803,^X1803
                                01C4  5752
                                01C4  5753 ; Control Store Address Comparison Mask
                                01C4  5754
                          3FFF  01C4  5755 3$:      .WORD   ^X3FFF
                                01C6  5756
                                01C6  5757 ; ROT Field Microorders
                                01C6  5758
                            3B  01C6  5759 4$:      .BYTE   ^X3B    ; ROT/OLIT0.PL_LIT
                            3D  01C7  5760          .BYTE   ^X3D    ; ROT/OLIT0.SL_LIT
                            3F  01C8  5761          .BYTE   ^X3F    ; ROT/OLIT0.PL43_WB
```

```
                        30  01C9  5762          .BYTE   ^X30     ; ROT/ZLIT0
                        31  01CA  5763          .BYTE   ^X31     ; ROT/ZLIT28
                        32  01CB  5764          .BYTE   ^X32     ; ROT/ZLIT24
                        33  01CC  5765          .BYTE   ^X33     ; ROT/ZLIT20
                        34  01CD  5766          .BYTE   ^X34     ; ROT/ZLIT16
                        35  01CE  5767          .BYTE   ^X35     ; ROT/ZLIT12
                        36  01CF  5768          .BYTE   ^X36     ; ROT/ZLIT8
                        37  01D0  5769          .BYTE   ^X37     ; ROT/ZLIT4
                        38  01D1  5770          .BYTE   ^X38     ; ROT/OLIT0
                        39  01D2  5771          .BYTE   ^X39     ; ROT/MINUS1
                        3A  01D3  5772          .BYTE   ^X3A     ; ROT/OLIT24
                        3C  01D4  5773          .BYTE   ^X3C     ; ROT/OLIT16
                        3E  01D5  5774          .BYTE   ^X3E     ; ROT/OLIT8
                            01D6  5775
                            01D6  5776 ; Reloadable Test Microinstruction
                            01D6  5777
                            01D6  5778 10$:
U 01, 5700,0840,EF64,DF00,0470,1800 01D6  5779 ;% 01   WB_RDM,RSRC/TEMP0,SPW/RLONG,ROT/OLIT0.PL_LIT,BUT/SRKSTA,NEXT/1800
                            01E1  5783
                            01E1  5784 BEGIN_CPU_DATA
                            0002  5785
                            0002  5786 DCS_DATA
                            0001  5787
U 00, 7700,C800,0364,0300,0470,1801 0001  5788 ;% 00   NOP
U 01, 5700,0840,EF64,DF00,0470,1800 000C  5792 ;% 01   WB_RDM,RSRC/TEMP0,SPW/RLONG,ROT/OLIT0.PL_LIT,BUT/SRKSTA,NEXT/1800
U 02, 7300,4800,0364,0300,0470,1803 0017  5796 ;% 02   STOP.CLOCK
                            0022  5800
                            0022  5801 RTEMP_DATA
                            0001  5802
                00000000    0001  5803          .LONG   0       ; INITIALIZES RTEMP0
                            0005  5804
                            0005  5805 END_CPU_DATA
                            01E1  5806
                            01E1  5807 END_TEST_DATA
                            01E1
                            01E1       ;-----------------------------------------------------------------------------
                            01E1  5808
                            01E1  5809 ENDTEST
```

```
                        001B       .SBTTL  "TEST   0A6     TEST BUT/CCBR0.SRKSTA0
                        001B  5811 ;*******************************************************************************
                        001B  5812 ;++
                        001B  5813 ;
                        001B  5814 ; FUNCTIONAL DESCRIPTION:
                        001B  5815 ;
                        001B  5816 ;       THIS TEST CHECKS THE BRANCHING LOGIC FOR CCBR0 AND SRKSTA0 LINES.
                        001B  5817 ;
                        001B  5818 ;
                        001B  5819 ; TEST ALGORITHM:
                        001B  5820 ;
                        001B  5821 ;       1.  CREATE A TEST LOOP OF 2
                        001B  5822 ;       2.  MODIFY THE MICRO CODE FOR THIS LOOP
                        001B  5823 ;       3.  LOAD THE MODIFIED MICRO CODE INTO THE DCS
                        001B  5824 ;       4.  LOAD A TEST PATTERN INTO THE WDREG
                        001B  5825 ;       5.  EXECUTE DCS PROGRAM
                        001B  5826 ;               a.  PUT A TEST PATTERN INTO THE Q REGISTER
                        001B  5827 ;               b.  PERFORM AN ADD WHILE SETTING THE ALUS FLAGS
                        001B  5828 ;               c.  LOAD THE PLATCH FROM THE RDM WDREG
                        001B  5829 ;               d.  PERFORM THE BRANCH ON CCBR0 AND SRKSTA0
                        001B  5830 ;       6.  TEST THE CS ADDRESS FOR THE CORRECT BRANCH
                        001B  5831 ;       7.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING LOOP COUNT
                        001B  5832 ;
                        001B  5833 ;
                        001B  5834 ; TEST PATTERNS:
                        001B  5835 ;
                        001B  5836 ;       ITERATION       Q + RTEMP               PLATCH          CS ADDRESS
                        001B  5837 ;           1       00000080 + 00000000          20              1801
                        001B  5838 ;           2       00000000 + 00000000          00              1802
                        001B  5839 ;
                        001B  5840 ;
                        001B  5841 ; LOGIC DESCRIPTION:
                        001B  5842 ;
                        001B  5843 ;       THIS TEST CHECKS THE DPM BRANCHING LOGIC WHICH GATES CCBR0 AND SRKSTA0
                        001B  5844 ;       ONTO CS ADDRESS BITS <1:0>.  ALL FAILURES SHOULD BE CONFINED TO THE
                        001B  5845 ;       DISCRETE BRANCHING LOGIC ON DPM.
                        001B  5846 ;
                        001B  5847 ;
                        001B  5848 ; ASSUMPTIONS:
                        001B  5849 ;
                        001B  5850 ;       THIS TEST ASSUMES THAT THE PREVIOUS BRANCHING TESTS HAVE ALL PASSED.
                        001B  5851 ;
                        001B  5852 ;
                        001B  5853 ; ERROR DESCRIPTION:
                        001B  5854 ;
                        001B  5855 ;       EXPECTED CS ADDRESS
                        001B  5856 ;       RECEIVED CS ADDRESS
                        001B  5857 ;       LOOP COUNT I
                        001B  5858 ;
                        001B  5859 ;--
                        001B  5860 ;*******************************************************************************
                        001B  5861
                        001B  5862
                        001B  5863         INITIALIZE                              ;INIT THE CPU
                        001D  5864         LOOP            I,1,2                   ;CREATE A TEST LOOP OF 2
```

```
                          0024  5865          LDFIELD         4$,I,B,5$,34,39          ;MODIFY MICRO CODE FOR LOOP
                          0035  5866          LOADDCS         5$,,01,1                 ;LOAD MICRO CODE INTO DCS
                          003C  5867          LOADREG         WDREG,1$,I,L             ;TEST PATTERN TO WDREG
                          0044  5868          ERRLOOP                                  ;ERROR LOOP POINT
                          0046  5869          FETCH           0                        ;START DCS PROGRAM
                          004B  5870          BRSTCLK         5                        ;END DCS PROGRAM
                          004F  5871          CMPREGMSK       CSTRP,2$,I,3$,,W,        ;TEST CS ADDRESS
                          005B  5872          IFERROR                                  ;INCORRECT BRANCH ADDRESS
                          0060  5873          ENDLOOP         I                        ;END TEST LOOP I
                          0063  5874          SKIP                                     ;GO TO NEXT TEST
                          006A  5875
                          006A  5876
                          006A  5877 BEGIN_TEST_DATA
                          006A
                          006A               ;-------------------------------------------------------------------------
                          006A  5878
                 00000020 006A  5879 1$:      .LONG    ^X00000020                     ;PLATCH DATA
                 00000000 006E  5880          .LONG    ^X00000000                     ;...
                     1801 0072  5881 2$:      .WORD    ^X1801                         ;CS BRANCH ADDRESS
                     1802 0074  5882          .WORD    ^X1802                         ;...
                     3FFF 0076  5883 3$:      .WORD    ^X3FFF                         ;MASK FOR CMPREGMSK PSEUDO OP
                    00 01 0078  5884 4$:      .BYTE    01,00                          ;RSRC FIELD
                          007A  5885 5$:
U 01, 7700,C800,5BE5,0304,0470,1802 007A  5886 ;% 01    Q_R[TEMP1]                     ;TEST DATA TO Q REGISTER
                          0085  5890
                          0085  5891
                          0085  5892 BEGIN_CPU_DATA
                          0002  5893
                          0002  5894 DCS_DATA
                          0001  5895
U 00, 7700,C800,0364,0300,0470,1801 0001  5896 ;% 00    NOP
U 01, 7700,C800,5BE5,0304,0470,1802 000C  5900 ;% 01    Q_R[TEMP1]                     ;TEST DATA TO Q REGISTER
U 02, 7701,8800,0390,0001,8C70,1803 0017  5904 ;% 02    SIZE[BYTE],ALU/A+B+CI,MUX/R.Q,RSRC/TEMP0,CCMISC/ALUS_SIGND.CCBR_ALUS
                          0022  5908                                                 ;SET CCBR0
U 03, 5700,8800,BF64,0300,0470,1804 0022  5909 ;% 03    ROT/SL.PL_WB,WB_RDM            ;LOAD PLATCH
U 04, 7700,4800,A364,DB01,8470,1800 002D  5913 ;% 04    BUT/CCBR0.SRKSTA0,CC/NOP.CCBR_ALUS,ROTSRK/PL.EQ.0.SIGN.120,NEXT/1800
                          0038  5917                                                 ;BRANCH ON CCBR0 & SRKSTA0
U 05, 7300,4800,0364,0300,0470,1806 0038  5918 ;% 05    NOP,STOP.CLOCK                 ;STOP THE CPU CLOCK
                          0043  5922
                          0043  5923 RTEMP_DATA
                          0001  5924
                 00000000 0001  5925          .LONG    ^X00000000                     ;RTEMP0
                 00000080 0005  5926          .LONG    ^X00000080                     ;RTEMP1
                          0009  5927
                          0009  5928 END_CPU_DATA
                          0085  5929
                          0085  5930
                          0085  5931
                          0085  5932 END_TEST_DATA
                          0085
                          0085               ;-------------------------------------------------------------------------
                          0085  5933 ENDTEST
```

```
0013       .SBTTL  "TEST   0A7     TEST BUT/BUTXB
0013  5935 ;*******************************************************************************
0013  5936 ;++
0013  5937 ;
0013  5938 ; FUNCTIONAL DESCRIPTION:
0013  5939 ;
0013  5940 ;       THIS TEST CHECKS THE DPM BRANCHING LOGIC WHICH GATES PREVIOUS
0013  5941 ;       DESTINATION INHIBIT ONTO CS ADDRESS BIT <0>.
0013  5942 ;
0013  5943 ;
0013  5944 ; TEST ALGORITHM:
0013  5945 ;
0013  5946 ;       SUBTEST #1
0013  5947 ;       1.  LOAD MICRO CODE FOR THIS SUBTEST
0013  5948 ;       2.  EXECUTE DCS PROGRAM
0013  5949 ;               a.  BRANCH ON PREV DEST INH L
0013  5950 ;       3.  TEST THE VBUS FOR GEN DEST INH L
0013  5951 ;       4.  IF NO ERROR TEST THE CS ADDRESS FOR THE CORRECT BRANCH
0013  5952 ;
0013  5953 ;       SUBTEST #2
0013  5954 ;       1.  LOAD MICRO CODE FOR THIS SUBTEST
0013  5955 ;       2.  LOAD THE FIRST MICRO INSTRUCTION AND HOLD MICRO ADDRESS INHIBIT
0013  5956 ;       3.  LOAD 1800 INTO THE CSAMR REGISTER
0013  5957 ;       4.  SET PARCHK AND MICRO ADDRESS INHIBIT IN THE DCSCR REGISTER
0013  5958 ;       5.  ALLOW THE CLOCK TO FREE RUN
0013  5959 ;               a.  TURN MEMORY MANAGEMENT ON
0013  5960 ;               b.  LOAD ZERO INTO THE VA REGISTER
0013  5961 ;               c.  INVALIDATE THE TRNASLATION BUFFER
0013  5962 ;               d.  LOAD THE PC WITH ZERO TO FORCE PREFETCH
0013  5963 ;               e.  READ THE EXECUTION BUFFER TO RTEMP3 (CAUSES MICRO TRAP)
0013  5964 ;               f.  BRANCH THE CS ADDRESS ON PREV DEST INH L
0013  5965 ;       6.  CLEAR PARCHK AND MICRO ADDRESS INHIBIT
0013  5966 ;       7.  TEST THE VBUS FOR GEN DEST INH L
0013  5967 ;       8.  IF NO ERROR TEST THE CS ADDRESS FOR THE CORRECT BRANCH
0013  5968 ;       9.  IF NO ERROR GO TO THE NEXT TEST
0013  5969 ;
0013  5970 ;
0013  5971 ; TEST PATTERNS:
0013  5972 ;
0013  5973 ;       NONE
0013  5974 ;
0013  5975 ;
0013  5976 ; LOGIC DESCRIPTION:
0013  5977 ;
0013  5978 ;       THIS TEST USES THE DPM BRANCHING LOGIC THAT GATES PREVIOUS DESTINATION
0013  5979 ;       INHIBIT ONTO CS ADDRESS BIT <0>.  SUBTEST #1 IS PRETTY STRAIGHTFORWARD,
0013  5980 ;       BUT SUBTEST #2 GETS A LITTLE MESSY DUE TO THE TIMING CONTRAINTS OF THE
0013  5981 ;       RDM.  SINCE THE SUBTEST MICRO TRAPS AND WE NEED TO LOOK AT PREV DEST
0013  5982 ;       INH ON THE NEXT MICRO CYCLE WE WERE UNABLE TO USE THE BRSTCLK PSEUDO
0013  5983 ;       OP.  WHAT THIS MEANS TO YOU IS THAT YOU WILL NOT BE ABLE TO SET A SCOPE
0013  5984 ;       SYNC POINT IN THIS SUBTEST.  HOWEVER, THERE IS NOT A GREAT DEAL OF
0013  5985 ;       LOGIC INVOLVED HERE AND MY BET IS THAT BY JUST LOOPING ON THE SUBTEST
0013  5986 ;       YOU SHOULD BE ABLE TO SCOPE OUT THE PROBLEM.
0013  5987 ;
0013  5988 ;
```

```
                        0013  5989 ; ASSUMPTIONS:
                        0013  5990 ;
                        0013  5991 ;       THIS TEST ASSUMES THE PREFETCH AND UTR MICRO TRAP LOGIC IS OPERATIONAL
                        0013  5992 ;
                        0013  5993 ;
                        0013  5994 ; ERROR DESCRIPTION:
                        0013  5995 ;
                        0013  5996 ;       SUBTEST #1
                        0013  5997 ;               FIRST IFERROR:
                        0013  5998 ;                       EXPECTED VBUS DATA (GEN DEST INH L)
                        0013  5999 ;                       RECEIVED VBUS DATA
                        0013  6000 ;
                        0013  6001 ;               SECOND IFERROR:
                        0013  6002 ;                       EXPECTED CS ADDRESS
                        0013  6003 ;                       RECEIVED CS ADDRESS
                        0013  6004 ;
                        0013  6005 ;       SUBTEST #2
                        0013  6006 ;               FIRST IFERROR:
                        0013  6007 ;                       EXPECTED VBUS DATA (GEN DEST INH)
                        0013  6008 ;                       RECEIVED VBUS DATA
                        0013  6009 ;
                        0013  6010 ;               SECOND IFERROR:
                        0013  6011 ;                       EXPECTED CS ADDRESS
                        0013  6012 ;                       RECEIVED CS ADDRESS
                        0013  6013 ;
                        0013  6014 ;       NOTE: VBUS DATA FORMAT IS:
                        0013  6015 ;               BITS <7:0> = VBUS BIT POSITION
                        0013  6016 ;               BITS <8> = BIT STATE
                        0013  6017 ;
                        0013  6018 ;--
                        0013  6019 ;*******************************************************************************
                        0013  6020 ;///////////////////////////////////////////////////////////////////////////////
                        0013  6021 ;+
                        0013  6022 ;SUBTEST #1
                        0013  6023 ;
                        0013  6024 ;       TEST WITHOUT MICROTRAP
                        0013  6025 ;-
                        0013  6026         SUBTEST                                         ;SUBTEST #1
                        0016
                        0016       ;////////////////////////////////////////////////////////////////////////
                        0016  6027         INITIALIZE                                      ;INIT THE CPU
                        0018  6028         LOADDCS         1$,,00,3                        ;LOAD MICRO CODE FOR SUBTEST
                        001F  6029         ERRLOOP                                         ;ERROR LOOP POINT
                        0021  6030         FETCH           0                               ;START DCS PROGRAM
                        0026  6031         BRSTCLK         2                               ;END DCS PROGRAM
                        002A  6032         CMPVBUS         4$                              ;TEST GEN DEST INH L
                        002F  6033         IFERROR         ,<UTR>,<MIC>                    ;SIGNAL INCORRECT
                        0036  6034         CMPREGMSK       CSTRP,2$,,3$,,W,                ;TEST CS ADDRESS
                        0042  6035         IFERROR         ,,<<DPM><UBI>>                  ;ADDRESS INCORRECT
                        0049  6036
                        0049  6037
                        0049  6038
                        0049  6039 ;///////////////////////////////////////////////////////////////////////////////
                        0049  6040 ;+
                        0049  6041 ;SUBTEST #2
```

```
                                0049  6042 ;
                                0049  6043 ;       TEST WITH MICROTRAP
                                0049  6044 ;-
                                0049  6045         SUBTEST                                         ;SUBTEST #2
                                004C
                                004C        ;////////////////////////////////////////////////////////////////////////
                                004C  6046         INITIALIZE                                      ;INIT THE CPU
                                004E  6047         LOADDCS         5$,,00,8                        ;LOAD MICRO CODE FOR SUBTEST
                                0055  6048         ERRLOOP                                         ;ERROR LOOP POINT
                                0057  6049         FETCH           0,INH                           ;SETUP FIRST MICRO INSTRUCTION
                                005C  6050         LOADREG         CSAMR,6$,,W                     ;LOAD 1800 INTO CSAMR REGISTER
                                0064  6051         LOADREG         DCSCR,7$,,B                     ;SET PARCHK AND UADDR INH
                                006C  6052         LOADREG         DCSCR,8$,,B                     ;ALLOW CLOCK TO FREE RUN
                                0074  6053         LOADREG         DCSCR,9$,,B                     ;REMOVE UADDR IHN
                                007C  6054         CMPVBUS         10$                             ;TEST GEN DEST INH L
                                0081  6055         IFERROR         ,<UTR>,<MIC>                    ;SIGNAL INCORRECT
                                0088  6056         CMPREGMSK       CSBUF,11$,,3$,,W,               ;TEST FOR CS BRANCH ADDRESS
                                0094  6057         IFERROR         ,,<<DPM><UBI>>                  ;INCORRECT BRANCH
                                009B  6058         SKIP                                            ;GO TO NEXT TEST
                                00A2  6059
                                00A2  6060
                                00A2  6061 BEGIN_TEST_DATA
                                00A2
                                00A2        ;-----------------------------------------------------------------------
                                00A2  6062
                                00A2  6063 1$:                                                     ;MICRO CODE SUBTEST #1
U 00, 7700,C800,0364,0300,0470,1801  00A2  6064 ;% 00    NOP
U 01, 7600,0800,0364,8300,0470,1800  00AD  6068 ;% 01    BUT/BUTXB,NEXT/1800,STROBE.V.BUS          ;BUT ON PREV DEST INH
U 02, 7300,4800,0364,0300,0470,1803  00B8  6072 ;% 02    NOP,STOP.CLOCK                            ;STOP THE CPU CLOCK
                                1801  00C3  6076 2$:      .WORD   ^X1801                          ;EXPECTED RESULTS
                                3FFF  00C5  6077 3$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                                  01  00C7  6078 4$:      .BYTE   ^X01                            ;VBUS RESULTS
                                      00C8  6079          VBUSDATA        <^X0C>,1                ;GEN DEST INH
                                      00C9  6080 5$:                                              ;MICRO CODE SUBTEST #2
U 00, 7700,C800,0364,0300,0470,1801  00C9  6081 ;% 00    NOP
U 01, 7700,C800,5BE4,0300,6870,1802  00D4  6085 ;% 01    MEMSCAR_R[TEMP0]                          ;ENABLE MME
U 02, 7700,8800,5BE4,0304,6070,1803  00DF  6089 ;% 02    MEMSCR_R[TEMP1]                           ;...
U 03, 7700,C800,5BE4,03D8,4A70,1804  00EA  6093 ;% 03    VA_R[ZERO]                                ;LOAD VA WITH ZERO
U 04, 7700,881B,5BE4,0308,5070,1805  00F5  6097 ;% 04    TB_R[TEMP2]                               ;INVALIDATE THE TB
U 05, 7700,C800,5BE4,03D8,4870,1806  0100  6101 ;% 05    PC_R[ZERO]                                ;FORCE PREFETCH
U 06, 7601,0857,5924,020E,0470,1807  010B  6105 ;% 06    R[TEMP3]_XB_PC_PC+4,STROBE.V.BUS          ;CAUSE A MICRO TRAP
U 07, 7300,0800,0364,8300,0470,1800  0116  6109 ;% 07    BUT/BUTXB,NEXT/1800,STOP.CLOCK            ;BUT ON PREV DEST INH
                                1800  0121  6113 6$:      .WORD   ^X1800                          ;DATA FOR CSAMR REGISTER
                                  F8  0123  6114 7$:      .BYTE   ^XF8                            ;PARCHK AND UADDR INH
                                  98  0124  6115 8$:      .BYTE   ^X98                            ;RUN CLOCK
                                  70  0125  6116 9$:      .BYTE   ^X70                            ;REMOVE PARCHK AND UADDR INH
                                  01  0126  6117 10$:     .BYTE   ^X01                            ;EXPECTED VBUS DATA
                                      0127  6118          VBUSDATA        <^X0C>,0                ;GEN DEST INH
                                1800  0128  6119 11$:     .WORD   ^X1800                          ;EXPECTED RESULTS
                                      012A  6120
                                      012A  6121
                                      012A  6122 BEGIN_CPU_DATA
                                      0002  6123
                                      0002  6124 RTEMP_DATA
                                      0001  6125
```

```
00000000  0001  6126          .LONG   ^X00000000
01000000  0005  6127          .LONG   ^X01000000
00000048  0009  6128          .LONG   ^X00000048
00000000  000D  6129          .LONG   ^X00000000
          0011  6130
          0011  6131 CACHE_DATA
          0001  6132
00000000  0001  6133          .LONG   ^X00000000
00000000  0005  6134          .LONG   ^X00000000
          0009  6135
          0009  6136 END_CPU_DATA
          012A  6137
          012A  6138
          012A  6139 END_TEST_DATA
          012A
          012A          ;-----------------------------------------------------------------------------
          012A  6140 ENDTEST
```

```
                        0020        .SBTTL  "TEST   0A8     TEST BUT/PSLTP AND BUT/PSLC
                        0020  6142 ;*******************************************************************************
                        0020  6143 ;++
                        0020  6144 ;
                        0020  6145 ; FUNCTIONAL DESCRIPTION:
                        0020  6146 ;
                        0020  6147 ;       THIS TEST CHECKS THE DPM BRANCHING LOGIC THAT GATES THE PSL TP AND C
                        0020  6148 ;       BITS ONTO CS ADDRESS BIT <0>.
                        0020  6149 ;
                        0020  6150 ;
                        0020  6151 ; TEST ALGORITHM:
                        0020  6152 ;
                        0020  6153 ;       1.  LOAD MICRO CODE FOR SUBTEST
                        0020  6154 ;       2.  CREATE A LOOP COUNT OF 2
                        0020  6155 ;       3.  LOAD PATTERN FOR PSL INTO WDREG
                        0020  6156 ;       4.  EXECUTE DCS PROGRAM
                        0020  6157 ;               a.  LOAD PSL WITH TEST PATTERN
                        0020  6158 ;               b.  BRANCH ON PSL TP(SUBTEST #1) OR C(SUBTEST #2) BIT
                        0020  6159 ;       5.  TEST CS BRANCH ADDRESS
                        0020  6160 ;       6.  IF NO ERROR GO TO STEP 3 FOR REMAINING TEST LOOP
                        0020  6161 ;
                        0020  6162 ;
                        0020  6163 ; TEST PATTERNS:
                        0020  6164 ;
                        0020  6165 ;                               ITERATION          PSL          CS ADDRESS
                        0020  6166 ;       SUBTEST #1
                        0020  6167 ;                                   1           00000000           1800
                        0020  6168 ;                                   2           40000000           1801
                        0020  6169 ;       SUBTEST #2
                        0020  6170 ;                                   1           00000000           1800
                        0020  6171 ;                                   2           00000001           1801
                        0020  6172 ;
                        0020  6173 ;
                        0020  6174 ; LOGIC DESCRIPTION:
                        0020  6175 ;
                        0020  6176 ;       THIS TEST EXECISES THE DISCRETE LOGIC THAT GATES THE PSL TP AND C
                        0020  6177 ;       BITS ONTO CS ADDRESS BIT <0>.
                        0020  6178 ;
                        0020  6179 ;
                        0020  6180 ; ERROR DESCRIPTION:
                        0020  6181 ;
                        0020  6182 ;       EXPECTED CS ADDRESS
                        0020  6183 ;       RECEIVED CS ADDRESS
                        0020  6184 ;       LOOP COUNT I
                        0020  6185 ;
                        0020  6186 ;--
                        0020  6187 ;*******************************************************************************
                        0020  6188 ;///////////////////////////////////////////////////////////////////////////////
                        0020  6189 ;+
                        0020  6190 ;SUBTEST #1
                        0020  6191 ;
                        0020  6192 ;       TEST BUT/PSLTP
                        0020  6193 ;-
                        0020  6194         SUBTEST                                         ;SUBTEST #1
                        0023
```

```
                                0023        ;////////////////////////////////////////////////////////////////////////////
                                0023        INITIALIZE                                          ;INIT THE CPU
                                0025        LOADDCS         1$,,02,1                            ;LOAD MICRO CODE FOR SUBTEST
                                002C        LOOP            I,1,2                               ;CREATE A TEST LOOP OF 2
                                0033        LOADREG         WDREG,2$,I,L                        ;TEST PATTERN TO WDREG
                                003B        ERRLOOP                                             ;ERROR LOOP POINT
                                003D        FETCH           0                                   ;START DCS PROGRAM
                                0042        BRSTCLK         3                                   ;END DCS PROGRAM
                                0046        CMPREGMSK       CSTRP,3$,I,4$,,W,                   ;TEST CS BRANCH ADDRESS
                                0052        IFERROR                                             ;ADDRESS INCORRECT
                                0057        ENDLOOP         I                                   ;END TEST LOOP
                                005A
                                005A
                                005A        ;////////////////////////////////////////////////////////////////////////////////
                                005A        ;+
                                005A        ;SUBTEST #2
                                005A        ;
                                005A        ;       TEST BUT/PSLC
                                005A        ;-
                                005A        SUBTEST                                             ;SUBTEST #2
                                005D
                                005D        ;////////////////////////////////////////////////////////////////////////////
                                005D        INITIALIZE                                          ;INIT THE CPU
                                005F        LOADDCS         5$,,02,1                            ;LOAD MICRO CODE FOR SUBTEST
                                0066        LOOP            I,1,2                               ;CREATE A TEST LOOP OF 2
                                006D        LOADREG         WDREG,6$,I,L                        ;TEST PATTERN TO WDREG
                                0075        ERRLOOP                                             ;ERROR LOOP POINT
                                0077        FETCH           0                                   ;START DCS PROGRAM
                                007C        BRSTCLK         3                                   ;END DCS PROGRAM
                                0080        CMPREGMSK       CSTRP,7$,I,4$,,W,                   ;TEST CS BRANCH ADDRESS
                                008C        IFERROR                                             ;ADDRESS INCORRECT
                                0091        ENDLOOP         I                                   ;END TEST LOOP
                                0094        SKIP                                                ;GO TO NEXT TEST
                                009B
                                009B
                                009B        BEGIN_TEST_DATA
                                009B
                                009B        ;-----------------------------------------------------------------------------
                                009B
                                009B        1$:                                                 ;MICRO CODE FOR SUBTEST #1
U 02, 7700,0800,0364,BF00,0470,1800  009B        ;% 02   BUT/PSLTP,NEXT/1800                         ;BRANCH ON TP BIT OF PSL
                        00000000  00A6        2$:     .LONG   ^X00000000                          ;DATA FOR PSL
                        40000000  00AA                .LONG   ^X40000000                          ;...
                            1800  00AE        3$:     .WORD   ^X1800                              ;RESULTS SUBTEST #1
                            1801  00B0                .WORD   ^X1801                              ;...
                            3FFF  00B2        4$:     .WORD   ^X3FFF                              ;MASK FOR CMPREGMSK PSEUDO OP
                                  00B4        5$:                                                 ;MICRO CODE FOR SUBTEST #2
U 02, 7700,0800,0364,9700,0470,1800  00B4        ;% 02   BUT/PSLC,NEXT/1800                          ;BRANCH ON C BIT OF PSL
                        00000000  00BF        6$:     .LONG   ^X00000000                          ;DATA FOR PSL
                        00000001  00C3                .LONG   ^X00000001                          ;...
                            1800  00C7        7$:     .WORD   ^X1800                              ;RESULTS SUBTEST #2
                            1801  00C9                .WORD   ^X1801                              ;...
                                  00CB
                                  00CB
                                  00CB        BEGIN_CPU_DATA
```

```
                                0002  6251
                                0002  6252 DCS_DATA
                                0001  6253
U 00. 7700,C800,0364,0300,0470,1801   0001  6254 ;% 00    NOP
U 01. 5700,4800,0364,0300,0070,1802   000C  6258 ;% 01    CCPSL/PSL_WB.CCBR_ALUS,WB_RDM          ;TEST PATTERN TO PSL
U 02. 7700,0800,0364,BF00,0470,1800   0017  6262 ;% 02    BUT/PSLTP,NEXT/1800                    ;PERFORM BRANCH
U 03. 7300,C800,0364,0300,0470,1804   0022  6266 ;% 03    NOP,STOP.CLOCK                         ;STOP THE CPU CLOCK
                                002D  6270
                                002D  6271 END_CPU_DATA
                                00CB  6272
                                00CB  6273
                                00CB  6274 END_TEST_DATA
                                00CB
                                00CB        ;-----------------------------------------------------------------------
                                00CB  6275 ENDTEST
```

```
0015          .SBTTL  "TEST   0A9     TEST BUT/REGMODE
0015  6277 ;*******************************************************************************
0015  6278 ;++
0015  6279 ;
0015  6280 ; FUNCTIONAL DESCRIPTION:
0015  6281 ;
0015  6282 ;       THIS TEST CHECKS THE DPM BRANCHING LOGIC THAT GATES REGMODE ONTO
0015  6283 ;       CS ADDRESS BIT <0>.
0015  6284 ;
0015  6285 ;
0015  6286 ; TEST ALGORITHM:
0015  6287 ;
0015  6288 ;       1.  CREATE A TEST LOOP OF 2
0015  6289 ;       2.  SAVE THE LOOP COUNT IN 1$
0015  6290 ;       3.  LOAD THE LOOP COUNT INTO THE WDREG
0015  6291 ;       4.  EXECUTE THE DCS PROGRAM
0015  6292 ;               a.  DISABLE ^P INTERRUPTS
0015  6293 ;               b.  MOVE THE LOOP COUNT INTO RTEMP2
0015  6294 ;               c.  PUT 2 IN THE PLATCH
0015  6295 ;               d.  ROTATE THE LOOP COUNT TO FORM A CACHE ADDRESS
0015  6296 ;               e.  LOAD THE ADDRESS INTO THE PC TO FORCE PREFETCH
0015  6297 ;               f.  PERFORM IRD1TEST TO LOAD THE INSTRUCTION REGISTER
0015  6298 ;               g.  BRANCH ON REGMODE
0015  6299 ;       5.  TEST THE VBUS FOR REGMODE
0015  6300 ;       6.  IF NO ERROR TEST THE CS ADDRESS FOR THE CORRECT BRANCH
0015  6301 ;       7.  IT NO ERROR GO TO STEP 2 FOR THE REMAINING LOOP COUNT
0015  6302 ;
0015  6303 ;
0015  6304 ; TEST PATTERNS:
0015  6305 ;
0015  6306 ;       ITERATION       INSTRUCTION     CS ADDRESS
0015  6307 ;           1             00000000        1800
0015  6308 ;           2             00005000        1801
0015  6309 ;
0015  6310 ;
0015  6311 ; LOGIC DESCRIPTION:
0015  6312 ;
0015  6313 ;       THIS TEST CHECKS THAT REGMODE CAN BE GATED ONTO CS ADDRESS BIT <0>.
0015  6314 ;       IF THE VBUS DATA IS BAD THAN THE PROBLEM IS MOST LIKELY IN THE IRD
0015  6315 ;       GATE ARRAY.  IF THE CS ADDRESS IS INCORRECT THAN YOU PROBABLY HAVE
0015  6316 ;       A FAILURE IN THE DPM BRANCHING LOGIC.
0015  6317 ;
0015  6318 ;
0015  6319 ; ASSUMPTIONS:
0015  6320 ;
0015  6321 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL
0015  6322 ;
0015  6323 ;
0015  6324 ; ERROR DESCRIPTION:
0015  6325 ;
0015  6326 ;       FIRST IFERROR
0015  6327 ;               EXPECTED VBUS DATA (DST REGMODE H)
0015  6328 ;               RECEIVED VBUS DATA
0015  6329 ;               LOOP COUNT I
0015  6330 ;                       (VBUS BITS <7:0> = VBUS BIT POSITION)
```

```
                                  0015  ;                        (VBUS BITS <8> = BIT STATE)
                                  0015  ;
                                  0015  ;       SECOND IFERROR
                                  0015  ;               EXPECTED CS ADDRESS
                                  0015  ;               RECEIVED CS ADDRESS
                                  0015  ;               LOOP COUNT I
                                  0015  ;
                                  0015  ;--
                                  0015  ;*******************************************************************************
                                  0015
                                  0015
                                  0015          INITIALIZE                                  ;INIT THE CPU
                                  0017          LOOP            I,1,2                       ;CREATE A TEST LOOP OF 2
                                  001E          SAVEINDEX       I,1$                        ;SAVE LOOP COUNT IN 1$
                                  0023          LOADREG         WDREG,1$,,L                 ;LOOP COUNT TO WDREG
                                  002B          ERRLOOP                                     ;ERROR LOOP POINT
                                  002D          FETCH           0                           ;START DCS PROGRAM
                                  0032          BRSTCLK         9                           ;END DCS PROGRAM
                                  0036          CMPVBUS         4$,I                        ;TEST DST RMODE SIGNAL
                                  003B          IFERROR         ,<IRD>,                     ;SIGNAL INCORRECT
                                  0041          CMPREGMSK       CSTRP,2$,I,3$,,W,           ;TEST CS BRANCH ADDRESS
                                  004D          IFERROR                                     ;ADDRESS INCORRECT
                                  0052          ENDLOOP         I                           ;END TEST LOOP
                                  0055          SKIP                                        ;GO TO NEXT TEST
                                  005C
                                  005C
                                  005C  BEGIN_TEST_DATA
                                  005C
                                  005C          ;-----------------------------------------------------------------------
                                  005C
                        00000000  005C  1$:     .LONG   ^X00000000                          ;LOOP COUNT STORAGE
                            1800  0060  2$:     .WORD   ^X1800                              ;CS BRANCH ADDRESSES
                            1801  0062          .WORD   ^X1801                              ;...
                            3FFF  0064  3$:     .WORD   ^X3FFF                              ;MASK FOR CMPREGMSK PSEUDO OP
                              01  0066  4$:     .BYTE   ^X1                                 ;VBUS DATA
                                  0067          VBUSDATA        <^X22>,0                    ;DST RMODE SIGNAL
                              01  0068          .BYTE   ^X1
                                  0069          VBUSDATA        <^X22>,1
                                  006A
                                  006A
                                  006A  BEGIN_CPU_DATA
                                  0002
                                  0002  DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 7700,4800,5BE4,0300,2470,1802  000C  ;% 01   WCTRL/LOADCRAR,WB_R[TEMP0]                 ;DISABLE ^P INTERRUPTS
U 02, 7700,0800,5BE4,0304,2C70,1803  0017  ;% 02   WCTRL/CONWRITE,WB_R[TEMP1]                 ;...
U 03, 5700,C840,0364,0308,0470,1804  0022  ;% 03   R[TEMP2]_RDM                                ;LOOP COUNT TO RTEMP2
U 04, 7700,D800,EF64,0301,0470,1805  002D  ;% 04   PL_[2]                                      ;PLATCH GETS 2
U 05, 7700,4800,8B71,0308,0470,1806  0038  ;% 05   Q_R[TEMP2].RL.P                             ;FORM CACHE ADDRESS
U 06, 7700,4800,03A4,03D8,4870,1807  0043  ;% 06   WCTRL/PC_WB,WB_Q                            ;FORCE PREFETCH
U 07, 7700,C800,0364,1700,0470,1808  004E  ;% 07   IRD1TEST                                    ;LOAD THE INSTRUCTION REGISTER
U 08, 7600,4800,0364,9300,0470,1800  0059  ;% 08   BUT/REGMODE,NEXT/1800,STROBE.V.BUS          ;PERFORM THE BRANCH
U 09, 7300,4800,0364,0300,0470,180A  0064  ;% 09   NOP,STOP.CLOCK                              ;STOP THE CPU CLOCK
                                  006F
```

```
                    006F  6414 RTEMP_DATA
                    0001  6415
00C00000            0001  6416          .LONG   ^X00C00000                      ;DISABLE ^P INTERRUPTS
00400000            0005  6417          .LONG   ^X00400000                      ;...
00000000            0009  6418          .LONG   ^X00000000                      ;LOOP COUNT STORAGE
                    000D  6419
                    000D  6420 CACHE_DATA
                    0001  6421
00000000            0001  6422          .LONG   ^X00000000                      ;INSTRUCTION TABLE
00005000            0005  6423          .LONG   ^X00005000
00000000            0009  6424          .LONG   ^X00000000
                    000D  6425
                    000D  6426 END_CPU_DATA
                    006A  6427
                    006A  6428
                    006A  6429 END_TEST_DATA
                    006A
                    006A          ;-----------------------------------------------------------------------
                    006A  6430 ENDTEST
```

```
001A        .SBTTL  "TEST   0AA     TEST BUT/CCBR1.INT-TS
001A  6432 ;*******************************************************************************
001A  6433 ;++
001A  6434 ;
001A  6435 ; FUNCTIONAL DESCRIPTION:
001A  6436 ;
001A  6437 ;       THIS TEST CHECKS THE BRANCHING LOGIC ON DPM THAT GATES CCBR1,
001A  6438 ;       INTERRUPT PENDING, AND TIMER SERVICE ONTO CS ADDRESS BITS <1:0>.
001A  6439 ;
001A  6440 ;
001A  6441 ; TEST ALGORITHM:
001A  6442 ;
001A  6443 ;       SUBTEST #1
001A  6444 ;       1.  LOAD MICRO CODE FOR THIS SUBTEST
001A  6445 ;       2.  EXECUTE DCS PROGRAM
001A  6446 ;               a.  LOAD A TEST PATTERN INTO THE Q REGISTER
001A  6447 ;               b.  ADD THE Q REGISTER TO RTEMP0 AND SET THE ALU STATE FLAGS
001A  6448 ;               c.  CLEAR THE INIR AND IICR REGISTERS
001A  6449 ;               d.  LOAD THE INIR WITH FFFF
001A  6450 ;               e.  SET THE TVP BIT IN TCSR
001A  6451 ;               f.  TRANSFER THE INIR TO IICR REGISTER
001A  6452 ;               g.  STEP THE IICR REGISTER BY ONE
001A  6453 ;               h.  BRANCH THE CS ADDRESS ON TIMER INTERRUPT
001A  6454 ;       3.  TEST THE CS ADDRESS FOR THE CORRECT BRANCH
001A  6455 ;
001A  6456 ;       SUBTEST #2
001A  6457 ;       1.  CREATE A TEST LOOP OF 2
001A  6458 ;       2.  MODIFY AND LOAD MICRO CODE FOR SUBTEST
001A  6459 ;       3.  EXECUTE DCS PROGRAM
001A  6460 ;               a.  LOAD A TEST PATTERN INTO THE Q REGISTER
001A  6461 ;               b.  ADD Q TO RTEMP0 AND SET THE ALU STATE FLAGS
001A  6462 ;               c.  SET THE T BIT IN TCSR
001A  6463 ;               d.  BRANCH ON TIMER SERVICE
001A  6464 ;       4.  TEST THE CS ADDRESS FOR THE CORRECT BRANCH
001A  6465 ;       5.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING LOOP COUNT
001A  6466 ;
001A  6467 ;
001A  6468 ; TEST PATTERNS:
001A  6469 ;
001A  6470 ;                               ITERATION       Q REG + RTEMP0          CS ADDRESS
001A  6471 ;       SUBTEST #1
001A  6472 ;                                   1           00000000 + 00000000         1801
001A  6473 ;       SUBTEST #2
001A  6474 ;                                   1           00000000 + 00000000         1801
001A  6475 ;                                   2           00000080 + 00000000         1802
001A  6476 ;
001A  6477 ;
001A  6478 ; LOGIC DESCRIPTION:
001A  6479 ;
001A  6480 ;       THIS TEST CHECKS ALL THE DISCRETE LOGIC ON DPM WHICH GATES CCBR1,
001A  6481 ;       INTERRUPT PENDING AND TIMER SERVICE ONTO CS ADDRESS BITS <1:0>.
001A  6482 ;       ANY FAILURES SHOULD BE CONFINED TO THIS AREA.
001A  6483 ;
001A  6484 ;
001A  6485 ; ASSUMPTIONS:
```

```
                        001A  6486 ;
                        001A  6487 ;       THIS TEST ASSUMES THAT THE ALU STATE FLAGS AND TIMER TESTS HAVE BEEN
                        001A  6488 ;       RUN SUCCESSFULLY.
                        001A  6489 ;
                        001A  6490 ;
                        001A  6491 ; ERROR DESCRIPTION:
                        001A  6492 ;
                        001A  6493 ;       EXPECTED CS ADDRESS
                        001A  6494 ;       RECEIVED CS ADDRESS
                        001A  6495 ;       LOOP COUNT I (SUBTEST #2 ONLY)
                        001A  6496 ;
                        001A  6497 ; --
                        001A  6498 ;*******************************************************************************
                        001A  6499 ;///////////////////////////////////////////////////////////////////////////////
                        001A  6500 ;+
                        001A  6501 ;SUBTEST #1
                        001A  6502 ;
                        001A  6503 ;       TEST WITH TIMER INTERRUPT
                        001A  6504 ;-
                        001A  6505         SUBTEST                                         ;SUBTEST #1
                        001D
                        001D       ;////////////////////////////////////////////////////////////////////////////
                        001D  6506         INITIALIZE                                      ;INIT THE CPU
                        001F  6507         LOADDCS         1$,,00,<^X0D>                   ;LOAD MICRO CODE FOR SUBTEST
                        0026  6508         ERRLOOP                                         ;ERROR LOOP POINT
                        0028  6509         FETCH           0                               ;START DCS PROGRAM
                        002D  6510         BRSTCLK         <^X0C>                          ;END DCS PROGRAM
                        0031  6511         CMPREGMSK       CSTRP,2$,,3$,,W,                ;TEST CS ADDRESS
                        003D  6512         IFERROR                                         ;INCORRECT BRANCH
                        0042  6513
                        0042  6514
                        0042  6515
                        0042  6516 ;///////////////////////////////////////////////////////////////////////////////
                        0042  6517 ;+
                        0042  6518 ;SUBTEST #2
                        0042  6519 ;
                        0042  6520 ;       TEST WITH TIMER SERVICE
                        0042  6521 ;-
                        0042  6522         SUBTEST                                         ;SUBTEST #2
                        0045
                        0045       ;////////////////////////////////////////////////////////////////////////////
                        0045  6523         INITIALIZE                                      ;INIT THE CPU
                        0047  6524         LOOP            I,1,2                           ;CREATE A TEST LOOP OF 2
                        004E  6525         LDFIELD         6$,I,B,5$,34,39                 ;MODIFY RSRC FIELD
                        005F  6526         LOADDCS         4$,,00,7                        ;LOAD MICRO CODE FOR SUBTEST
                        0066  6527         ERRLOOP                                         ;ERROR LOOP POINT
                        0068  6528         FETCH           0                               ;START DCS PROGRAM
                        006D  6529         BRSTCLK         6                               ;END DCS PROGRAM
                        0071  6530         CMPREGMSK       CSTRP,7$,I,3$,,W,               ;TEST CS ADDRESS
                        007D  6531         IFERROR                                         ;INCORRECT BRANCH
                        0082  6532         ENDLOOP         I                               ;END TEST LOOP
                        0085  6533         SKIP                                            ;GO TO NEXT TEST
                        008C  6534
                        008C  6535
                        008C  6536 BEGIN_TEST_DATA
```

```
                                 008C
                                 008C      ;-------------------------------------------------------------------------
                                 008C  6537
                                 008C  6538 1$:                                                    ;MICRO CODE SUBTEST #1
U 00. 7700,C800,0364,0300,0470,1801  008C  6539 ;* 00   NOP
U 01. 7700,8800,5BE5,0300,0470,1802  0097  6543 ;* 01   Q_R[TEMP0]                                 ;TEST DATA TO Q REGISTER
U 02. 7701,8800,0390,0001,8C70,1803  00A2  6547 ;* 02   SIZE[BYTE],ALU/A+B+CI,MUX/R.Q,RSRC/TEMP0,CCMISC/ALUS_SIGND.CCBR_ALUS
                                 00AD  6551                                                       ;SET ALU STATE FLAGS
U 03. 7700,0800,5BE4,0308,1470,1804  00AD  6552 ;* 03   WCTRL/INIR_WB,WB_R[TEMP2]                  ;CLEAR INIR REGISTER
U 04. 7700,4800,5BE4,030C,1670,1805  00B8  6556 ;* 04   WCTRL/TCSR_WB,WB_R[TEMP3]                  ;CLEAR IICR REGISTER
U 05. 7700,8800,5BE4,0310,1470,1806  00C3  6560 ;* 05   WCTRL/INIR_WB,WB_R[TEMP4]                  ;LOAD INIR REGISTER WITH FFFF
U 06. 7700,C800,5BE4,0314,1670,1807  00CE  6564 ;* 06   WCTRL/TCSR_WB,WB_R[TEMP5]                  ;SET TVP BIT IN TCSR REGISTER
U 07. 7700,C800,5BE4,0318,1670,1808  00D9  6568 ;* 07   WCTRL/TCSR_WB,WB_R[TEMP6]                  ;TRANSFER INIR TO IICR REGISTER
U 08. 7700,0800,5BE4,031C,1670,1809  00E4  6572 ;* 08   WCTRL/TCSR_WB,WB_R[TEMP7]                  ;STEP IICR BY ONE
U 09. 7700,4800,0364,0300,0470,180A  00EF  6576 ;* 09   NOP                                       ;ALLOW INTERRUPT TO SETUP
U 0A. 7700,C800,0364,0300,0470,180B  00FA  6580 ;* 0A   NOP                                       ;...
U 0B. 7700,0800,0364,B301,8470,1800  0105  6584 ;* 0B   BUT/CCBR1.INT-TS,NEXT/1800,CC/NOP.CCBR_ALUS
U 0C. 7300,C800,0364,0300,0470,180D  0110  6588 ;* 0C   NOP,STOP.CLOCK                             ;STOP THE CPU CLOCK
                                 011B  6592
                             1801  011B  6593 2$:      .WORD   ^X1801                            ;RESULTS SUBTEST #1
                             3FFF  011D  6594 3$:      .WORD   ^X3FFF                            ;MASK FOR CMPREGMSK PSEUDO OP
                                 011F  6595
                                 011F  6596 4$:                                                    ;MICRO CODE SUBTEST #2
U 00. 7700,C800,0364,0300,0470,1801  011F  6597 ;* 00   NOP
                                 012A  6601 5$:
U 01. 7700,8800,5BE5,0300,0470,1802  012A  6602 ;* 01   Q_R[TEMP0]                                 ;TEST DATA TO Q REGISTER
U 02. 7701,8800,0390,0001,8C70,1803  0135  6606 ;* 02   SIZE[BYTE],ALU/A+B+CI,MUX/R.Q,RSRC/TEMP0,CCMISC/ALUS_SIGND.CCBR_ALUS
                                 0140  6610                                                       ;SET ALU STATE FLAGS
U 03. 7700,8800,5BE4,0320,1670,1804  0140  6611 ;* 03   WCTRL/TCSR_WB,WB_R[TEMP8]                  ;SET T BIT IN TCSR REGISTER
U 04. 7700,4800,0364,0300,0470,1805  014B  6615 ;* 04   NOP                                       ;ALLOW TIMER SERVICE TO SETUP
U 05. 7700,0800,0364,B301,8470,1800  0156  6619 ;* 05   BUT/CCBR1.INT-TS,NEXT/1800,CC/NOP.CCBR_ALUS
U 06. 7300,C800,0364,0300,0470,1807  0161  6623 ;* 06   NOP,STOP.CLOCK                             ;STOP THE CPU CLOCK
                                 016C  6627
                               00  016C  6628 6$:      .BYTE   00                                ;RSRC LDFIELD TABLE
                               01  016D  6629          .BYTE   01
                             1801  016E  6630 7$:      .WORD   ^X1801                            ;RESULTS SUBTEST #2
                             1802  0170  6631          .WORD   ^X1802
                                 0172  6632
                                 0172  6633
                                 0172  6634 BEGIN_CPU_DATA
                                 0002  6635
                                 0002  6636 RTEMP_DATA
                                 0001  6637
                         00000000  0001  6638          .LONG   ^X00000000                        ;TEST DATA
                         00000080  0005  6639          .LONG   ^X00000080                        ;...
                         00000000  0009  6640          .LONG   ^X00000000                        ;CLEAR INIR
                         00100000  000D  6641          .LONG   ^X00100000                        ;CLEAR IICR
                         0000FFFF  0011  6642          .LONG   ^X0000FFFF                        ;INIR = FFFF
                         01000000  0015  6643          .LONG   ^X01000000                        ;TVP BIT SET
                         01500000  0019  6644          .LONG   ^X01500000                        ;TRANSFER INIR TO IICR
                         01640000  001D  6645          .LONG   ^X01640000                        ;STEP IICR
                         00100000  0021  6646          .LONG   ^X00100000                        ;CAUSE T BIT TO SET
                                 0025  6647
                                 0025  6648 END_CPU_DATA
                                 0172  6649
```

```
                        0172  6650
                        0172  6651 END_TEST_DATA
                        0172
                        0172       ;-----------------------------------------------------------------------------
                        0172  6652 ENDTEST
```

```
0014          .SBTTL  "TEST   0AB     TEST BUT/BCDCHK
0014  6654 ;*******************************************************************************
0014  6655 ;++
0014  6656 ;
0014  6657 ; FUNCTIONAL DESCRIPTION:
0014  6658 ;
0014  6659 ;       THIS TEST CHECKS THE ALP AND BRANCHING LOGIC THAT SUPPORTS THE BCD
0014  6660 ;       CHECK FUNCTION.
0014  6661 ;
0014  6662 ;
0014  6663 ; TEST ALGORITHM:
0014  6664 ;
0014  6665 ;       1.  CREATE A TEST LOOP J OF 8
0014  6666 ;       2.  MODIFY AND LOAD MICRO CODE ACCORDING TO LOOP J
0014  6667 ;       3.  CREATE A SECOND TEST LOOP I OF 16
0014  6668 ;       4.  SAVE LOOP COUNT I IN 1$
0014  6669 ;       5.  LOAD THE LOOP COUNT INTO THE WDREG
0014  6670 ;       6.  EXECUTE THE DCS PROGRAM
0014  6671 ;               a.  MOVE THE LOOP COUNT TO RTEMP0
0014  6672 ;               b.  LOAD THE RDM WHREG WITH A PATTERN TO REFLECT THE ALP UNDER-
0014  6673 ;                       TEST.
0014  6674 ;               c.  LOAD THE PLATCH WITH A ROTATE VALUE
0014  6675 ;               d.  ROTATE THE LOOP COUNT TO THE TEST POSITION AND SAVE IN THE
0014  6676 ;                       D REGISTER.
0014  6677 ;               e.  PERFORM A BCD CHECK ON THE VALUE IN THE D REGISTER
0014  6678 ;       7.  TEST THE CS ADDRESS FOR THE CORRECT BRANCH
0014  6679 ;       8.  IF NO ERROR BRANCH AROUND THE ERROR CALLOUT AND CONTINUE WITH
0014  6680 ;               STEP 4 UNTIL I IS EXHAUSTED. THEN GO TO STEP 2 UNTIL J IS
0014  6681 ;               EXHAUSTED.
0014  6682 ;       9.  IF THERE IS AN ERROR USE THE PATTERN IN THE WHREG TO CALL OUT THE
0014  6683 ;               CORRECT ALP GATE ARRAY.
0014  6684 ;
0014  6685 ;
0014  6686 ; TEST PATTERNS:
0014  6687 ;
0014  6688 ;       ITERATION (I)           TEST DATA       CS ADDRESS
0014  6689 ;           1                   00000000            1800
0014  6690 ;           2                   00000001            1800
0014  6691 ;           3                   00000002            1800
0014  6692 ;           4                   00000003            1800
0014  6693 ;           5                   00000004            1800
0014  6694 ;           6                   00000005            1800
0014  6695 ;           7                   00000006            1800
0014  6696 ;           8                   00000007            1800
0014  6697 ;           9                   00000008            1800
0014  6698 ;           A                   00000009            1800
0014  6699 ;           B                   0000000A            1801
0014  6700 ;           C                   0000000B            1801
0014  6701 ;           D                   0000000C            1801
0014  6702 ;           E                   0000000D            1801
0014  6703 ;           F                   0000000E            1801
0014  6704 ;          10                   0000000F            1801
0014  6705 ;
0014  6706 ;       NOTE: AS LOOP J INCREMENTS THE TEST PATTERN SIMPLY IS SHIFTED LEFT
0014  6707 ;               TO THE NEXT NIBBLE.
```

```
                        0014  6708 ;
                        0014  6709 ;
                        0014  6710 ; LOGIC DESCRIPTION:
                        0014  6711 ;
                        0014  6712 ;       THIS TEST CHECKS THAT THE ALP GATE ARRAYS IN CONJUCTION WITH THE CLA
                        0014  6713 ;       GATE ARRAY CAN RECOGNIZE AN INVALID BCD DIGIT (# > 9).  ALSO TESTED
                        0014  6714 ;       IS THE BRANCHING LOGIC ON DPM WHICH GATES BCDCHK ONTO CS ADDRESS
                        0014  6715 ;       BIT <0>.
                        0014  6716 ;
                        0014  6717 ;
                        0014  6718 ; ERROR DESCRIPTION:
                        0014  6719 ;
                        0014  6720 ;       EXPECTED DATA (WILL ALWAYS BE 0)
                        0014  6721 ;       RECEIVED DATA (NOT MEANINGFUL; USED TO FORCE CORRECT ALP CALLOUT)
                        0014  6722 ;       LOOP COUNT I
                        0014  6723 ;       LOOP COUNT J
                        0014  6724 ;
                        0014  6725 ;--
                        0014  6726 ;*******************************************************************************
                        0014  6727
                        0014  6728
                        0014  6729          INITIALIZE                              ;INIT THE CPU
                        0016  6730          LOOP            J,1,8                   ;TEST LOOP OF 8
                        001D  6731          LDFIELD         7$,J,B,5$,34,39         ;CHANGE RSRC FIELD
                        002E  6732          LDFIELD         8$,J,W,6$,31,39         ;CHANGE LITRL FIELD
                        003F  6733          LOADDCS         5$,,02,2                ;LOAD MICRO CODE FOR THIS LOOP
                        0046  6734          LOOP            I,1,16                  ;CREATE A TEST LOOP OF 16
                        004D  6735          SAVEINDEX       I,1$                    ;SAVE LOOP COUNT
                        0052  6736          LOADREG         WDREG,1$,,L             ;MOVE LOOP COUNT TO WDREG
                        005A  6737          ERRLOOP                                 ;ERROR LOOP POINT
                        005C  6738          FETCH           0                       ;START DCS PROGRAM
                        0061  6739          BRSTCLK         6                       ;END DCS PROGRAM
                        0065  6740          CMPREGMSK       CSTRP,2$,I,3$,,W,       ;TEST CS ADDRESS
                        0071  6741          SKIP            10$,NOERROR             ;SKIP ON NO ERROR
                        0078  6742          CMPREG          WHREG,4$,,L             ;TEST ALP DATA FOR CALLOUT
                        0081  6743          IFERROR         ,<<ALP><CLA>>           ;PRINT GATE ARRAYS
                        0088  6744 10$:     ENDLOOP         I                       ;END TEST LOOP I
                        008B  6745          ENDLOOP         J                       ;END TEST LOOP J
                        008E  6746          SKIP                                    ;GO TO NEXT TEST
                        0095  6747
                        0095  6748
                        0095  6749 BEGIN_TEST_DATA
                        0095
                        0095        ;-------------------------------------------------------------------------
                        0095  6750
                00000000  0095  6751 1$:      .LONG   ^X00000000                     ;LOOP COUNT STORAGE
                    1800  0099  6752 2$:      .WORD   ^X1800                         ;RESULTS TABLE
                    1800  009B  6753          .WORD   ^X1800
                    1800  009D  6754          .WORD   ^X1800
                    1800  009F  6755          .WORD   ^X1800
                    1800  00A1  6756          .WORD   ^X1800
                    1800  00A3  6757          .WORD   ^X1800
                    1800  00A5  6758          .WORD   ^X1800
                    1800  00A7  6759          .WORD   ^X1800
                    1800  00A9  6760          .WORD   ^X1800
```

```
                          1800  00AB  6761          .WORD   ^X1800
                          1801  00AD  6762          .WORD   ^X1801
                          1801  00AF  6763          .WORD   ^X1801
                          1801  00B1  6764          .WORD   ^X1801
                          1801  00B3  6765          .WORD   ^X1801
                          1801  00B5  6766          .WORD   ^X1801
                          1801  00B7  6767          .WORD   ^X1801
                          3FFF  00B9  6768 3$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                      00000000  00BB  6769 4$:      .LONG   ^X00000000                      ;COMPARE DATA
                                00BF  6770 5$:
U 02, 6700,0800,5BE4,0304,0470,1803   00BF  6771 ;% 02   RDM_R[TEMP1]                       ;SELECTS ALP FOR CALLOUT
                                00CA  6775 6$:
U 03, 7700,1800,EF64,0300,0470,1804   00CA  6776 ;% 03   PL_[0]                            ;ROTATES DATA TO CORRECT BITS
          08 07 06 05 04 03 02 01     00D5  6780 7$:      .BYTE   01,02,03,04,05,06,07,08     ;RSRC LDFIELD DATA
              000C 0008 0004 0000     00DD  6781 8$:      .WORD   ^X0000,^X0004,^X0008,^X000C ;LITRL LDFIELD DATA
              001C 0018 0014 0010     00E5  6782          .WORD   ^X0010,^X0014,^X0018,^X001C ;...
                                00ED  6783
                                00ED  6784 BEGIN_CPU_DATA
                                0002  6785
                                0002  6786 DCS_DATA
                                0001  6787
U 00, 7700,C800,0364,0300,0470,1801   0001  6788 ;% 00   NOP
U 01, 5700,8840,0364,0300,0470,1802   000C  6792 ;% 01   R[TEMP0]_RDM                       ;TEST PATTERN TO RTEMP0
U 02, 6700,0800,5BE4,0304,0470,1803   0017  6796 ;% 02   RDM_R[TEMP1]                       ;ERROR CALLOUT DATA TO RDM
U 03, 7700,1800,EF64,0300,0470,1804   0022  6800 ;% 03   PL_[0]                            ;# OF BITS TO SHIFT LEFT
U 04, 7700,4800,87E6,03D8,0470,1805   002D  6804 ;% 04   ROT/RL.MM.P,MSRC/TEMP0,RSRC/ZERO,ALU/OR,MUX/R.S,DQ1/D_WX
U 05, 7700,0800,0214,9BD8,0470,1800   0038  6808 ;% 05   BUT/BCDCHK,ALU/A+B+CI.BCD,MUX/D.R1,RSRC/ZERO,NEXT/1800
                                0043  6812                                              ;PERFORM BCK CHECK
U 06, 7300,C800,0364,0300,0470,1807   0043  6813 ;% 06   NOP,STOP.CLOCK                  ;STOP THE CPU CLOCK
                                004E  6817
                                004E  6818 RTEMP_DATA
                                0001  6819
                      00000000  0001  6820          .LONG   ^X00000000                  ;TEST PATTERN STORAGE
                      00000001  0005  6821          .LONG   ^X00000001                  ;ALP 1 CALLOUT
                      00000010  0009  6822          .LONG   ^X00000010                  ;ALP 2 CALLOUT
                      00000100  000D  6823          .LONG   ^X00000100                  ;ALP 3 CALLOUT
                      00001000  0011  6824          .LONG   ^X00001000                  ;ALP 4 CALLOUT
                      00010000  0015  6825          .LONG   ^X00010000                  ;ALP 5 CALLOUT
                      00100000  0019  6826          .LONG   ^X00100000                  ;ALP 6 CALLOUT
                      01000000  001D  6827          .LONG   ^X01000000                  ;ALP 7 CALLOUT
                      10000000  0021  6828          .LONG   ^X10000000                  ;ALP 8 CALLOUT
                                0025  6829
                                0025  6830 END_CPU_DATA
                                00ED  6831
                                00ED  6832
                                00ED  6833 END_TEST_DATA
                                00ED
                                00ED        ;-----------------------------------------------------------------------------
                                00ED  6834 ENDTEST
```

```
                        0012        .SBTTL  "TEST   0AC     TEST BUT/FPS3
                        0012  6836 ;*******************************************************************************
                        0012  6837 ;++
                        0012  6838 ;
                        0012  6839 ; FUNCTIONAL DESCRIPTION:
                        0012  6840 ;
                        0012  6841 ;       THIS TEST CHECKS FOR FRONT PANEL LOCKED AND ACLO VIA THE DPM
                        0012  6842 ;       BRANCHING LOGIC.
                        0012  6843 ;
                        0012  6844 ;
                        0012  6845 ; TEST ALGORITHM:
                        0012  6846 ;
                        0012  6847 ;       SUBTEST #1
                        0012  6848 ;       1.  EXECUTE DCS PROGRAM
                        0012  6849 ;               a.  BRANCH ON FRONT PANEL LOCKED AND ACLO
                        0012  6850 ;       2.  TEST THE CS ADDRESS FOR NOT FRONT PANEL LOCKED AND NOT ACLO
                        0012  6851 ;       3.  IF NO ERROR GO TO SUBTEST #2
                        0012  6852 ;       4.  IF ERROR DECIDE WHICH ONE
                        0012  6853 ;
                        0012  6854 ;       SUBTEST #2
                        0012  6855 ;       1.  LOAD THE FIRST MICRO INSTRUCTION
                        0012  6856 ;       2.  ASSERT ACLO FROM THE RDM
                        0012  6857 ;       3.  EXECUTE DCS PROGRAM
                        0012  6858 ;               a.  BRANCH ON ACLO
                        0012  6859 ;       4.  CLEAR ACLO
                        0012  6860 ;       5.  TEST CS ADDRESS FOR ACLO
                        0012  6861 ;       6.  IF NO ERROR GO TO THE NEXT TEST
                        0012  6862 ;
                        0012  6863 ;
                        0012  6864 ; TEST PATTERNS:
                        0012  6865 ;
                        0012  6866 ;       NONE
                        0012  6867 ;
                        0012  6868 ;
                        0012  6869 ; LOGIC DESCRIPTION:
                        0012  6870 ;
                        0012  6871 ;       THIS TEST USES THE BUT MICRO ORDER TO DETERMINE IF THE FRONT PANEL
                        0012  6872 ;       SWITCH IS LOCKED AND WHETHER ACLO IS OPERATIONAL.  IF THE FRONT PANEL
                        0012  6873 ;       SWITCH IS LOCKED IT MUST BE AN ERROR CONDITION IN THE LOGIC, SINCE
                        0012  6874 ;       MICRO DIAGNOSTICS CANNOT RUN IN THAT STATE.
                        0012  6875 ;
                        0012  6876 ;
                        0012  6877 ; ERROR DESCRIPTION:
                        0012  6878 ;
                        0012  6879 ;       SUBTEST #1
                        0012  6880 ;               EXPECTED CS ADDRESS
                        0012  6881 ;               RECEIVED CS ADDRESS
                        0012  6882 ;       or
                        0012  6883 ;               FRONT PANEL LOCKED
                        0012  6884 ;
                        0012  6885 ;       SUBTEST #2
                        0012  6886 ;               EXPECTED CS ADDRESS
                        0012  6887 ;               RECEIVED CS ADDRESS
                        0012  6888 ;
                        0012  6889 ;--
```

```
                     0012  6890 ;*******************************************************************************
                     0012  6891 ;///////////////////////////////////////////////////////////////////////////////
                     0012  6892 ;+
                     0012  6893 ;SUBTEST #1
                     0012  6894 ;
                     0012  6895 ;       NOT ACLO AND FP LOCK
                     0012  6896 ;-
                     0012  6897         SUBTEST                                         ;SUBTEST #1
                     0015
                     0015       ;////////////////////////////////////////////////////////////////////////
                     0015  6898         INITIALIZE                                      ;INIT THE CPU
                     0017  6899         ERRLOOP                                         ;ERROR LOOP POINT
                     0019  6900         FETCH           0                               ;START DCS PROGRAM
                     001E  6901         BRSTCLK         3                               ;END DCS PROGRAM
                     0022  6902         CMPREGMSK       CSTRP,1$,,2$,,W,                ;TEST THE CS BRANCH ADDRESS
                     002E  6903         SKIP            10$,NOERROR                     ;GO TO SUBTEST #2
                     0035  6904         CMPREGMSK       CSTRP,3$,,2$,,W,                ;TEST FOR FRONT PANEL LOCKED
                     0041  6905         SKIP            11$,NOERROR                     ;SKIP IF LOCKED
                     0048  6906         IFERROR         ,,<UBI>                         ;ACLO IS ASSERTED
                     004E  6907         SKIP            10$                             ;GO TO SUBTEST #2
                     0055  6908 11$:     PRINT_S         8$                              ;FRONT PANEL LOCKED
                     0059  6909
                     0059  6910
                     0059  6911 ;///////////////////////////////////////////////////////////////////////////////
                     0059  6912 ;+
                     0059  6913 ;SUBTEST #2
                     0059  6914 ;
                     0059  6915 ;       TEST WITH ACLO
                     0059  6916 ;-
                     0059  6917 10$:     SUBTEST                                         ;SUBTEST #2
                     005C
                     005C       ;////////////////////////////////////////////////////////////////////////
                     005C  6918         INITIALIZE                                      ;INIT THE CPU
                     005E  6919         ERRLOOP                                         ;ERROR LOOP POINT
                     0060  6920         FETCH           0                               ;START DCS PROGRAM
                     0065  6921         LOADREG         RDCTRL,4$,,B                    ;SET ACLO
                     006D  6922         BRSTCLK         3                               ;END DCS PROGRAM
                     0071  6923         LOADREG         RDCTRL,5$,,B                    ;CLEAR ACLO
                     0079  6924         CMPREGMSK       CSTRP,6$,,7$,,W,                ;TEST CS BRANCH ADDRESS
                     0085  6925         IFERROR                                         ;INCORRECT BRANCH
                     008A  6926         SKIP                                            ;GO TO NEXT TEST
                     0091  6927
                     0091  6928
                     0091  6929 BEGIN_TEST_DATA
                     0091
                     0091       ;-------------------------------------------------------------------------
                     0091  6930
                1800 0091  6931 1$:      .WORD   ^X1800                                  ;NOT ACLO AND FRONT PANEL LOCK
                3FFF 0093  6932 2$:      .WORD   ^X3FFF                                  ;MASK FOR CMPREGMSK PSEUDO OP
                1801 0095  6933 3$:      .WORD   ^X1801                                  ;FRONT PANEL LOCKED
                  A8 0097  6934 4$:      .BYTE   ^XA8                                    ;SET ACLO
                  28 0098  6935 5$:      .BYTE   ^X28                                    ;CLEAR ACLO
                1802 0099  6936 6$:      .WORD   ^X1802                                  ;BRANCH FOR ACLO
                3FFE 009B  6937 7$:      .WORD   ^X3FFE                                  ;MASK FOR CMPREGMSK PSEUDO OP
4E 41 50 20 54 4E 4F 52 46 0A 0D 00' 009D  6938 8$:      .ASCIC  <13><10>/FRONT PANEL LOCKED/<13><10>
```

```
   0A 0D 44 45 4B 43 4F 4C 20 4C 45  00A9
                                 16  009D
                                     00B4  6939
                                     00B4  6940 BEGIN_CPU_DATA
                                     0002  6941
                                     0002  6942 DCS_DATA
                                     0001  6943
U 00. 7700,C800,0364,0300,0470,1801  0001  6944 ;% 00   NOP
U 01. 7700,C800,0364,0300,0470,1802  000C  6948 ;% 01   NOP
U 02. 7700,0800,0364,CB00,0470,1800  0017  6952 ;% 02   BUT/FPS3,NEXT/1800                    ;BRANCH ON ACLO AND FRONT PANEL
U 03. 7300,C800,0364,0300,0470,1804  0022  6956 ;% 03   NOP,STOP,CLOCK                       ;STOP THE CPU CLOCK
                                     002D  6960
                                     002D  6961 END_CPU_DATA
                                     00B4  6962
                                     00B4  6963
                                     00B4  6964 END_TEST_DATA
                                     00B4
                                     00B4       ;-----------------------------------------------------------------------
                                     00B4  6965 ENDTEST
```

```
                        0014        .SBTTL  "TEST   0AD     TEST BUT/NO.FPA
                        0014  6967 ;*******************************************************************************
                        0014  6968 ;++
                        0014  6969 ;
                        0014  6970 ; FUNCTIONAL DESCRIPTION:
                        0014  6971 ;
                        0014  6972 ;       THIS TEST CHECKS FOR THE PRESENCE OF THE FLOATING POINT ACCELERATOR
                        0014  6973 ;       OPTION.
                        0014  6974 ;
                        0014  6975 ;
                        0014  6976 ; TEST ALGROITHM:
                        0014  6977 ;
                        0014  6978 ;       1.  EXECUTE DCS PROGRAM
                        0014  6979 ;               a.  BRANCH ON FPA PRESENT
                        0014  6980 ;       2.  TEST CS ADDRESS FOR FPA PRESENT
                        0014  6981 ;       3.  PRINT FPA MESSAGE
                        0014  6982 ;
                        0014  6983 ;
                        0014  6984 ; TEST PATTERNS:
                        0014  6985 ;
                        0014  6986 ;       NONE
                        0014  6987 ;
                        0014  6988 ;
                        0014  6989 ; LOGIC DESCRIPTION:
                        0014  6990 ;
                        0014  6991 ;       THIS TEST USES THE DPM BRANCHING LOGIC TO DETERMINE WHETHER THE
                        0014  6992 ;       FPA IS PRESENT.  SINCE EITHER CASE IS POSSIBLE THE ONLY
                        0014  6993 ;       ALTERNATIVE IS TO PRINT WHAT THE TEST SEES AND ALLOW THE USER
                        0014  6994 ;       TO DECIDE WHETHER IT IS CORRECT.
                        0014  6995 ;
                        0014  6996 ;
                        0014  6997 ; ASSUMPTIONS:
                        0014  6998 ;
                        0014  6999 ;       THIS TEST ASSUMES THE RDM POWER-UP TESTS HAVE RUN.
                        0014  7000 ;
                        0014  7001 ;
                        0014  7002 ; ERROR DESCRIPTION:
                        0014  7003 ;
                        0014  7004 ;       FPA MESSAGE
                        0014  7005 ;          or
                        0014  7006 ;       EXPECTED CS ADDRESS
                        0014  7007 ;       RECEIVED CS ADDRESS
                        0014  7008 ;               (INDICATES FPA NOT PRESENT AND ENABLED)
                        0014  7009 ;
                        0014  7010 ;--
                        0014  7011 ;*******************************************************************************
                        0014  7012
                        0014  7013
                        0014  7014
                        0014  7015          INITIALIZE                              ;INIT THE CPU
                        0016  7016          ERRLOOP                                 ;ERROR LOOP POINT
                        0018  7017          FETCH           0                       ;START DCS PROGRAM
                        001D  7018          BRSTCLK         2                       ;END DCS PROGRAM
                        0021  7019          CMPREGMSK       CSTRP,1$,,2$,,W,        ;TEST CS ADDRESS
                        002D  7020          SKIP            10$,ONERROR             ;SKIP IF NOT 1800
```

```
                                0034  7021          PRINT_S         3$                      ;FPA PRESENT AND ENABLED
                                0038  7022          SKIP                                    ;GO TO NEXT TEST
                                003F  7023 10$:     CMPREGMSK       CSTRP,4$,,2$,,W,        ;TEST CS ADDRESS
                                004B  7024          SKIP            20$,ONERROR             ;SKIP IF NOT 1801
                                0052  7025          PRINT_S         5$                      ;FPA PRESENT AND NOT ENABLED
                                0056  7026          SKIP                                    ;GO TO NEXT TEST
                                005D  7027 20$:     CMPREGMSK       CSTRP,6$,,2$,,W         ;TEST CS ADDRESS
                                0069  7028          SKIP            30$,ONERROR             ;SKIP IF NOT 1803
                                0070  7029          PRINT_S         7$                      ;FPA NOT PRESENT
                                0074  7030          SKIP                                    ;GO TO NEXT TEST
                                007B  7031 30$:     IFERROR                                 ;FPA NOT PRESENT AND ENABLED?
                                0080  7032          SKIP                                    ;GO TO NEXT TEST
                                0087  7033
                                0087  7034
                                0087  7035 BEGIN_TEST_DATA
                                0087
                                0087          ;-----------------------------------------------------------------------
                                0087  7036
                          1800  0087  7037 1$:       .WORD   ^X1800                          ;CS ADDRESS
                          3FFF  0089  7038 2$:       .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
45 53 45 52 50 20 41 50 46 0A 0D 00'  008B  7039 3$:       .ASCIC  <13><10>/FPA PRESENT AND ENABLED/<13><10>
4C 42 41 4E 45 20 44 4E 41 20 54 4E   0097
                  0A 0D 44 45         00A3
                           1B         008B
                          1801  00A7  7040 4$:       .WORD   ^X1801                          ;CS ADDRESS
45 53 45 52 50 20 41 50 46 0A 0D 00'  00A9  7041 5$:       .ASCIC  <13><10>/FPA PRESENT AND NOT ENABLED/<13><10>
45 20 54 4F 4E 20 44 4E 41 20 54 4E   00B5
         0A 0D 44 45 4C 42 41 4E      00C1
                              1F      00A9
                          1803  00C9  7042 6$:       .WORD   ^X1803                          ;CS ADDRESS
50 20 54 4F 4E 20 41 50 46 0A 0D 00'  00CB  7043 7$:       .ASCIC  <13><10>/FPA NOT PRESENT/<13><10>
         0A 0D 54 4E 45 53 45 52      00D7
                              13      00CB
                                00DF  7044
                                00DF  7045
                                00DF  7046
                                00DF  7047 BEGIN_CPU_DATA
                                0002  7048
                                0002  7049 DCS_DATA
                                0001  7050
U 00, 7700,C800,0364,0300,0470,1801   0001  7051 ;% 00    NOP
U 01, 7700,4800,0364,C300,0470,1800   000C  7055 ;% 01    BUT/NO.FPA,NEXT/1800            ;BRANCH ON FPA PRESENT
U 02, 7300,4800,0364,0300,0470,1803   0017  7059 ;% 02    NOP,STOP.CLOCK                  ;STOP THE CPU CLOCK
                                0022  7063
                                0022  7064 END_CPU_DATA
                                00DF  7065
                                00DF  7066
                                00DF  7067 END_TEST_DATA
                                00DF
                                00DF          ;-----------------------------------------------------------------------
                                00DF  7068 ENDTEST
```

```
                        0012        .SBTTL  "TEST   0AE     TEST BUT/FPS2
                        0012  7070 ;*******************************************************************************
                        0012  7071 ;++
                        0012  7072 ;
                        0012  7073 ; FUNCTIONAL DESCRIPTION:
                        0012  7074 ;
                        0012  7075 ;       THIS TEST CHECKS THE FRONT PANEL BOOT DEVICE SWITCH AND PRINTS
                        0012  7076 ;       ITS POSITION.
                        0012  7077 ;
                        0012  7078 ;
                        0012  7079 ; TEST ALGORITHM:
                        0012  7080 ;
                        0012  7081 ;       1.  EXECUTE DCS PROGRAM
                        0012  7082 ;               a.  BRANCH ON BOOT DEVICE SWITCH
                        0012  7083 ;       2.  TEST THE CS ADDRESS AND PRINT THE RESULTS
                        0012  7084 ;
                        0012  7085 ;
                        0012  7086 ; TEST PATTERNS:
                        0012  7087 ;
                        0012  7088 ;       NONE
                        0012  7089 ;
                        0012  7090 ;
                        0012  7091 ; LOGIC DESCRIPTION:
                        0012  7092 ;
                        0012  7093 ;       SINCE ANY POSITION OF THE BOOT SWITCH IS LEGAL THE USER WILL HAVE TO
                        0012  7094 ;       DECIDE WHETHER THE PRINTOUT IS CORRECT.  LOOPING ON TEST AND TURNING
                        0012  7095 ;       THE BOOT DEVICE SWITCH WILL ALLOW YOU TO TEST ALL POSITIONS OF THE
                        0012  7096 ;       SWITCH.
                        0012  7097 ;
                        0012  7098 ;
                        0012  7099 ; ERROR DESCRIPTION:
                        0012  7100 ;
                        0012  7101 ;       BOOT DEVICE *
                        0012  7102 ;
                        0012  7103 ;--
                        0012  7104 ;*******************************************************************************
                        0012  7105
                        0012  7106
                        0012  7107          INITIALIZE                              ;INIT THE CPU
                        0014  7108          ERRLOOP                                 ;ERROR LOOP POINT
                        0016  7109          FETCH           0                       ;START DCS PROGRAM
                        001B  7110          BRSTCLK         2                       ;END DCS PROGRAM
                        001F  7111          CMPREGMSK       CSTRP,1$,,4$,,W,        ;TEST CS ADDRESS
                        002B  7112          SKIP            10$,ONERROR             ;NOT DEVICE D
                        0032  7113          PRINT_S         5$                      ;BOOT DEVICE D
                        0036  7114          SKIP                                    ;GO TO NEXT TEST
                        003D  7115 10$:     CMPREGMSK       CSTRP,2$,,4$,,W,        ;TEST CS ADDRESS
                        0049  7116          SKIP            20$,ONERROR             ;NOT DEVICE C
                        0050  7117          PRINT_S         6$                      ;BOOT DEVICE C
                        0054  7118          SKIP                                    ;GO TO NEXT TEST
                        005B  7119 20$:     CMPREGMSK       CSTRP,3$,,4$,,W,        ;TEST CS ADDRESS
                        0067  7120          SKIP            30$,ONERROR             ;NOT DEVICE B
                        006E  7121          PRINT_S         7$                      ;BOOT DEVICE B
                        0072  7122          SKIP                                    ;GO TO NEXT TEST
                        0079  7123 30$:     PRINT_S         8$                      ;BOOT DEVICE A
```

```
                                   007D          SKIP                                    ;GO TO NEXT TEST
                                   0084
                                   0084
                                   0084  BEGIN_TEST_DATA
                                   0084
                                   0084      ;-----------------------------------------------------------------------
                                   0084
                             1800  0084  1$:     .WORD   ^X1800                          ;BOOT DEVICE D
                             1801  0086  2$:     .WORD   ^X1801                          ;BOOT DEVICE C
                             1802  0088  3$:     .WORD   ^X1802                          ;BOOT DEVICE B
                             3FFF  008A  4$:     .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
49 56 45 44 20 54 4F 4F 42 0A 0D 00' 008C  5$:     .ASCIC  <13><10>/BOOT DEVICE D/<13><10>
                 0A 0D 44 20 45 43  0098
                                11  008C
49 56 45 44 20 54 4F 4F 42 0A 0D 00' 009E  6$:     .ASCIC  <13><10>/BOOT DEVICE C/<13><10>
                 0A 0D 43 20 45 43  00AA
                                11  009E
49 56 45 44 20 54 4F 4F 42 0A 0D 00' 00B0  7$:     .ASCIC  <13><10>/BOOT DEVICE B/<13><10>
                 0A 0D 42 20 45 43  00BC
                                11  00B0
49 56 45 44 20 54 4F 4F 42 0A 0D 00' 00C2  8$:     .ASCIC  <13><10>/BOOT DEVICE A/<13><10>
                 0A 0D 41 20 45 43  00CE
                                11  00C2
                                   00D4
                                   00D4
                                   00D4  BEGIN_CPU_DATA
                                   0002
                                   0002  DCS_DATA
                                   0001
U 00, 7700,C800,0364,0300,0470,1801 0001  ;% 00   NOP
U 01, 7700,4800,0364,CF00,0470,1800 000C  ;% 01   BUT/FPS2,NEXT/1800                     ;BUT ON BOOT DEVICE
U 02, 7300,4800,0364,0300,0470,1803 0017  ;% 02   NOP,STOP.CLOCK                         ;STOP THE CPU CLOCK
                                   0022
                                   0022  END_CPU_DATA
                                   00D4
                                   00D4
                                   00D4  END_TEST_DATA
                                   00D4
                                   00D4      ;-----------------------------------------------------------------------
                                   00D4  ENDTEST
```

```
                        0012        .SBTTL  "TEST   0AF     TEST BUT/FPS1
                        0012  7162 ;*******************************************************************************
                        0012  7163 ;++
                        0012  7164 ;
                        0012  7165 ; FUNCTIONAL DESCRIPTION:
                        0012  7166 ;
                        0012  7167 ;       THIS TEST CHECKS THE BRANCH ON CONSOLE HALT AND FRONT PANEL
                        0012  7168 ;       POWER ON ACTION SWITCH.
                        0012  7169 ;
                        0012  7170 ;
                        0012  7171 ; TEST ALGORITHM:
                        0012  7172 ;
                        0012  7173 ;       SUBTEST #1
                        0012  7174 ;       1.  CREATE A TEST LOOP OF 2
                        0012  7175 ;       2.  LOAD MICRO CODE TO CLEAR/SET CONSOLE HALT
                        0012  7176 ;       3.  EXECUTE DCS PROGRAM
                        0012  7177 ;               a.  CLEAR/SET ^P INTERRUPT
                        0012  7178 ;               b.  BRANCH ON CONSOLE HALT SIGNAL
                        0012  7179 ;       4.  TEST CS ADDRESS FOR CORRECT BRANCH
                        0012  7180 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING LOOP
                        0012  7181 ;
                        0012  7182 ;       SUBTEST #2
                        0012  7183 ;       1.  EXECUTE DCS PROGRAM
                        0012  7184 ;               a.  SET ^P INTERRUPT
                        0012  7185 ;               b.  BRANCH ON FRONT PANEL POWER ON ACTION SWITCH
                        0012  7186 ;       2.  PRINT "POWER ON ACTION IS "
                        0012  7187 ;       3.  TEST CS ADDRESS AND PRINT SWITCH POSITION
                        0012  7188 ;
                        0012  7189 ;
                        0012  7190 ; TEST PATTERNS:
                        0012  7191 ;
                        0012  7192 ;       NONE
                        0012  7193 ;
                        0012  7194 ;
                        0012  7195 ; LOGIC DESCRIPTION:
                        0012  7196 ;
                        0012  7197 ;       SUBTEST #1 MASKS OUT THE POWER ON ACTION SWITCH AND ONLY TESTS THE
                        0012  7198 ;       CONDITION OF THE CONSOLE HALT SIGNAL.  SUBTEST #2 DOES THE INVERSE.
                        0012  7199 ;       BY LOOPING ON TEST AND TURNING THE POWER ON ACTION SWITCH YOU CAN
                        0012  7200 ;       TEST ALL POSITIONS OF THE SWITCH.
                        0012  7201 ;
                        0012  7202 ;
                        0012  7203 ; ASSUMPTIONS:
                        0012  7204 ;
                        0012  7205 ;       THIS TEST ASSUMES THE CONSOLE HALT INTERRUPT HAS BEEN TESTED
                        0012  7206 ;
                        0012  7207 ;
                        0012  7208 ; ERROR DESCRIPTION:
                        0012  7209 ;
                        0012  7210 ;       SUBTEST #1
                        0012  7211 ;               EXPECTED CS ADDRESS
                        0012  7212 ;               RECEIVED CS ADDRESS
                        0012  7213 ;               LOOP COUNT I
                        0012  7214 ;
                        0012  7215 ;       SUBTEST #2
```

```
                     0012  7216 ;                    POWER ON ACTION IS *
                     0012  7217 ;
                     0012  7218 ;--
                     0012  7219 ;*******************************************************************************
                     0012  7220 ;///////////////////////////////////////////////////////////////////////////////
                     0012  7221 ;+
                     0012  7222 ;SUBTEST #1
                     0012  7223 ;
                     0012  7224 ;       TEST BRANCH ON CONSOLE HALT
                     0012  7225 ;-
                     0012  7226         SUBTEST                                         ;SUBTEST #1
                     0015
                     0015       ;////////////////////////////////////////////////////////////////////////
                     0015  7227         INITIALIZE                                      ;INIT THE CPU
                     0017  7228         LOOP            I,1,2                           ;CREATE A TEST LOOP OF 2
                     001E  7229         LOADDCS         3$,I,02,1                       ;LOAD MICRO CODE FOR THIS LOOP
                     0025  7230         ERRLOOP                                         ;ERROR LOOP POINT
                     0027  7231         FETCH           0                               ;START DCS PROGRAM
                     002C  7232         BRSTCLK         4                               ;END DCS PROGRAM
                     0030  7233         CMPREGMSK       CSTRP,1$,I,2$,,W,               ;TEST CS ADDRESS
                     003C  7234         IFERROR                                         ;INCORRECT BRANCH ADDRESS
                     0041  7235         ENDLOOP         I                               ;END TEST LOOP
                     0044  7236
                     0044  7237
                     0044  7238 ;///////////////////////////////////////////////////////////////////////////////
                     0044  7239 ;+
                     0044  7240 ;SUBTEST #2
                     0044  7241 ;
                     0044  7242 ;       TEST BRANCH ON FRONT PANEL SWITCH 1
                     0044  7243 ;-
                     0044  7244         SUBTEST                                         ;SUBTEST #2
                     0047
                     0047       ;////////////////////////////////////////////////////////////////////////
                     0047  7245         INITIALIZE                                      ;INIT THE CPU
                     0049  7246         ERRLOOP                                         ;ERROR LOOP POINT
                     004B  7247         FETCH           0                               ;START DCS PROGRAM
                     0050  7248         BRSTCLK         4                               ;END DCS PROGRAM
                     0054  7249         PRINT_S         12$                             ;POWER ON ACTION IS
                     0058  7250         CMPREGMSK       CSTRP,4$,,7$,,W,                ;TEST CS ADDRESS
                     0064  7251         SKIP            20$,ONERROR                     ;NOT R/BOOT
                     006B  7252         PRINT_S         8$                              ;R/BOOT
                     006F  7253         SKIP                                            ;GO TO NEXT TEST
                     0076  7254 20$:    CMPREGMSK       CSTRP,5$,,7$,,W,                ;TEST CS ADDRESS
                     0082  7255         SKIP            30$,ONERROR                     ;NOT R/HALT
                     0089  7256         PRINT_S         9$                              ;R/HALT
                     008D  7257         SKIP                                            ;GO TO NEXT TEST
                     0094  7258 30$:    CMPREGMSK       CSTRP,6$,,7$,,W,                ;TEST CS ADDRESS
                     00A0  7259         SKIP            40$,ONERROR                     ;NOT BOOT
                     00A7  7260         PRINT_S         10$                             ;BOOT
                     00AB  7261         SKIP                                            ;GO TO NEXT TEST
                     00B2  7262 40$:    PRINT_S         11$                             ;HALT
                     00B6  7263         SKIP                                            ;GO TO NEXT TEST
                     00BD  7264
                     00BD  7265
                     00BD  7266 BEGIN_TEST_DATA
```

```
                                    00BD
                                    00BD        ;-----------------------------------------------------------------------
                                    00BD  7267
                              1804  00BD  7268 1$:     .WORD   ^X1804                          ;RESULTS SUBTEST #1
                              1800  00BF  7269         .WORD   ^X1800                          ;...
                              3FF4  00C1  7270 2$:     .WORD   ^X3FF4                          ;MASK FOR CMPREGMSK PSEUDO OP
                                    00C3  7271 3$:                                             ;MICRO CODE SUBTEST #1
U 02, 7700,0800,5BE4,0304,2C70,1803 00C3  7272 ;% 02   WCTRL/CONWRITE,WB_R[TEMP1]              ;CLEAR CONSOLE HALT
U 02, 7700,0800,5BE4,0308,2C70,1803 00CE  7276 ;% 02   WCTRL/CONWRITE,WB_R[TEMP2]              ;SET CONSOLE HALT
                              1800  00D9  7280 4$:     .WORD   ^X1800                          ;R/BOOT
                              1801  00DB  7281 5$:     .WORD   ^X1801                          ;R/HALT
                              1802  00DD  7282 6$:     .WORD   ^X1802                          ;BOOT
                              3FF3  00DF  7283 7$:     .WORD   ^X3FF3                          ;MASK FOR CMPREGMSK PSEUDO OP
4F 4F 42 2D 54 52 41 54 53 45 52 00' 00E1  7284 8$:     .ASCIC  /RESTART-BOOT/<13><10>
                          0A 0D 54  00ED
                                0E  00E1
4C 41 48 2D 54 52 41 54 53 45 52 00' 00F0  7285 9$:     .ASCIC  /RESTART-HALT/<13><10>
                          0A 0D 54  00FC
                                0E  00F0
              0A 0D 54 4F 4F 42 00' 00FF  7286 10$:    .ASCIC  /BOOT/<13><10>
                                06  00FF
              0A 0D 54 4C 41 48 00' 0106  7287 11$:    .ASCIC  /HALT/<13><10>
                                06  0106
20 4E 4F 20 52 45 57 4F 50 0A 0D 00' 010D  7288 12$:    .ASCIC  <13><10>/POWER ON ACTION IS /
      20 53 49 20 4E 4F 49 54 43 41  0119
                                15  010D
                                    0123  7289
                                    0123  7290
                                    0123  7291 BEGIN_CPU_DATA
                                    0002  7292
                                    0002  7293 DCS_DATA
                                    0001  7294
U 00, 7700,C800,0364,0300,0470,1801 0001  7295 ;% 00   NOP
U 01, 7700,4800,5BE4,0300,2470,1802 000C  7299 ;% 01   WCTRL/LOADCRAR,WB_R[TEMP0]              ;CLEAR/SET CONSOLE HALT
U 02, 7700,0800,5BE4,0304,2C70,1803 0017  7303 ;% 02   WCTRL/CONWRITE,WB_R[TEMP1]              ;...
U 03, 7700,0800,0364,D300,0470,1800 0022  7307 ;% 03   BUT/FPS1,NEXT/1800                      ;BUT ON HALT AND FRONT PANEL
U 04, 7300,4800,0364,0300,0470,1805 002D  7311 ;% 04   NOP,STOP.CLOCK                          ;STOP THE CPU CLOCK
                                    0038  7315
                                    0038  7316 RTEMP_DATA
                                    0001  7317
                          00C00000  0001  7318         .LONG   ^X00C00000                      ;CONSOLE HALT FUNCTION
                          00400000  0005  7319         .LONG   ^X00400000                      ;CLEAR ^P
                          00900000  0009  7320         .LONG   ^X00900000                      ;SET ^P
                                    000D  7321
                                    000D  7322 END_CPU_DATA
                                    0123  7323
                                    0123  7324
                                    0123  7325 END_TEST_DATA
                                    0123
                                    0123        ;-----------------------------------------------------------------------
                                    0123  7326 ENDTEST
```

```
                        0016        .SBTTL  "TEST   0B0     BUT/SPASTA TEST 1
                        0016  7328 ;*******************************************************************************
                        0016  7329 ;++
                        0016  7330 ;
                        0016  7331 ; FUNCTIONAL DESCRIPTION:
                        0016  7332 ;
                        0016  7333 ;       THIS IS ONE IN A SERIES OF 4 TESTS THAT CHECK THE SPASTA BRANCHING
                        0016  7334 ;       LOGIC.
                        0016  7335 ;
                        0016  7336 ;
                        0016  7337 ; TEST ALGORITHM:
                        0016  7338 ;
                        0016  7339 ;       1.  CREATE A TEST LOOP I = 16
                        0016  7340 ;       2.  SAVE THE LOOP COUNT IN 1$
                        0016  7341 ;       3.  MOVE THE LOOP COUNT TO THE WDREG
                        0016  7342 ;       4.  EXECUTE THE DCS PROGRAM
                        0016  7343 ;               a.  MOVE WDREG DATA TO RNUM WHILE BRANCHING ON SPASTA
                        0016  7344 ;               b.  STOP THE CPU CLOCK
                        0016  7345 ;       5.  TEST THE CS ADDRESS FOR THE CORRECT BRANCH
                        0016  7346 ;       6.  IF NO ERROR GO TO STEP 2 UNTIL LOOP I IS EXHAUSTED
                        0016  7347 ;
                        0016  7348 ;
                        0016  7349 ; TEST PATTERNS:
                        0016  7350 ;
                        0016  7351 ;       ITERATION       CS ADDRESS
                        0016  7352 ;           1               1803
                        0016  7353 ;           2               1803
                        0016  7354 ;           3               1803
                        0016  7355 ;           4               1803
                        0016  7356 ;           5               1803
                        0016  7357 ;           6               1802
                        0016  7358 ;           7               1802
                        0016  7359 ;           8               1802
                        0016  7360 ;           9               1800
                        0016  7361 ;           A               1800
                        0016  7362 ;           B               1800
                        0016  7363 ;           C               1800
                        0016  7364 ;           D               1800
                        0016  7365 ;           E               1800
                        0016  7366 ;           F               1802
                        0016  7367 ;          10               1802
                        0016  7368 ;
                        0016  7369 ;
                        0016  7370 ; LOGIC DESCRIPTION:
                        0016  7371 ;
                        0016  7372 ;       THIS TEST EXERCISES THE LOGIC INTERNAL TO THE SPA GATE ARRAY
                        0016  7373 ;       THAT IS RESPONSIBLE FOR THE SPA ST1 AND ST0 BITS DURING
                        0016  7374 ;       THE MICRO ORDER MSRC/RNUM_WBUS.  ANY FAILURES SHOULD BE CONFINED
                        0016  7375 ;       TO THE SPA GATE ARRAY OR THE DISCRETE BRANCHING LOGIC ON DPM.
                        0016  7376 ;
                        0016  7377 ;
                        0016  7378 ; ASSUMPTIONS:
                        0016  7379 ;
                        0016  7380 ;       THIS TEST ASSUMES THE WBUS IS OPERATIONAL.
                        0016  7381 ;
```

```
                                0016  ;
                                0016  ; ERROR DESCRIPTION:
                                0016  ;
                                0016  ;       EXPECTED CS ADDRESS
                                0016  ;       RECEIVED CS ADDRESS
                                0016  ;       LOOP COUNT I
                                0016  ;
                                0016  ;--
                                0016  ;*******************************************************************************
                                0016
                                0016
                                0016          INITIALIZE                                  ;INIT THE CPU
                                0018          LOOP            I,1,<^X10>                  ;CREATE A TEST LOOP OF 16
                                001F          SAVEINDEX       I,1$                        ;SAVE LOOP COUNT IN 1$
                                0024          LOADREG         WDREG,1$,,L                 ;LOAD LOOP COUNT INTO WDREG
                                002C          ERRLOOP                                     ;ERROR LOOP POINT
                                002E          FETCH           0                           ;START DCS PROGRAM
                                0033          BRSTCLK         2                           ;END DCS PROGRAM
                                0037          CMPREGMSK       CSTRP,2$,I,3$,,W,           ;TEST CS ADDRESS
                                0043          IFERROR         ,<SPA>,                     ;INCORRECT BRANCH ADDRESS
                                0049          ENDLOOP         I                           ;END TEST LOOP I
                                004C          SKIP                                        ;GO TO NEXT TEST
                                0053
                                0053
                                0053  BEGIN_TEST_DATA
                                0053
                                0053          ;-----------------------------------------------------------------------
                                0053
                        00000000  0053  1$:     .LONG   ^X00000000                          ;LOOP COUNT STORAGE
                            1803  0057  2$:     .WORD   ^X1803                              ;RESULTS TABLE
                            1803  0059          .WORD   ^X1803
                            1803  005B          .WORD   ^X1803
                            1803  005D          .WORD   ^X1803
                            1803  005F          .WORD   ^X1803
                            1802  0061          .WORD   ^X1802
                            1802  0063          .WORD   ^X1802
                            1802  0065          .WORD   ^X1802
                            1800  0067          .WORD   ^X1800
                            1800  0069          .WORD   ^X1800
                            1800  006B          .WORD   ^X1800
                            1800  006D          .WORD   ^X1800
                            1800  006F          .WORD   ^X1800
                            1800  0071          .WORD   ^X1800
                            1802  0073          .WORD   ^X1802
                            1802  0075          .WORD   ^X1802
                            3FFF  0077  3$:     .WORD   ^X3FFF                              ;MASK FOR CMPREGMSK PSEUDO OP
                                0079
                                0079
                                0079  BEGIN_CPU_DATA
                                0002
                                0002  DCS_DATA
                                0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 5700,481D,0364,BB00,0470,1800  000C  ;% 01   MSRC/RNUM_WBUS,WB_RDM,RSRC/TEMP0,BUT/SPASTA,NEXT/1800
U 02, 7300,4800,0364,0300,0470,1803  0017  ;% 02   NOP,STOP.CLOCK
```

```
0022  7444
0022  7445 END_CPU_DATA
0079  7446
0079  7447
0079  7448 END_TEST_DATA
0079
0079          ;-----------------------------------------------------------------------------
0079  7449 ENDTEST
```

```
0016        .SBTTL  "TEST   0B1     BUT/SPASTA TEST 2
0016  7451 ;*******************************************************************************
0016  7452 ;++
0016  7453 ;
0016  7454 ; FUNCTIONAL DESCRIPTION:
0016  7455 ;
0016  7456 ;       THIS IS ONE IN A SERIES OF 4 TESTS THAT CHECK THE SPASTA BRANCHING
0016  7457 ;       LOGIC.
0016  7458 ;
0016  7459 ;
0016  7460 ; TEST ALGORITHM:
0016  7461 ;
0016  7462 ;       1.  CREATE A TEST LOOP I = 2
0016  7463 ;       2.  LOAD MICRO CODE INTO DCS ACCORDING TO I
0016  7464 ;       3.  EXECUTE DCS PROGRAM
0016  7465 ;               a.  CLEAR THE STACK POINTER TO ZERO
0016  7466 ;               b.  PUSH THE STACK
0016  7467 ;               c.  CLEAR THE STACK POINTER AGAIN
0016  7468 ;               d.  BRANCH ON THE VALUE OF RBS<6>
0016  7469 ;               e.  STOP THE CPU CLOCK
0016  7470 ;       4.  TEST THE CS ADDRESS FOR THE CORRECT BRANCH
0016  7471 ;       5.  IF NO ERROR GO TO STEP 2 FOR NEXT LOOP
0016  7472 ;
0016  7473 ;
0016  7474 ; TEST PATTERNS:
0016  7475 ;
0016  7476 ;       ITERATION       CS ADDRESS
0016  7477 ;           1               1802
0016  7478 ;           2               1800
0016  7479 ;
0016  7480 ;
0016  7481 ; LOGIC DESCRIPTION:
0016  7482 ;
0016  7483 ;       THIS TEST EXERCISES THE LOGIC INTERNAL TO THE SPA GATE ARRAY THAT
0016  7484 ;       IS RESPONSIBLE FOR THE SPA ST1 AND ST0 BITS DURING THE MICRO
0016  7485 ;       ORDER MSRC/READRBS.  ANY FAILURES SHOULD BE CONFINED TO THE SPA
0016  7486 ;       GATE ARRAY OF THE DISCRETE BRANCHING LOGIC ON DPM.
0016  7487 ;
0016  7488 ;
0016  7489 ; ASSUMPTIONS:
0016  7490 ;
0016  7491 ;       THIS TEST ASSUMES THE WBUS IS OPERATIONAL.
0016  7492 ;
0016  7493 ;
0016  7494 ; ERROR DESCRIPTION:
0016  7495 ;
0016  7496 ;       EXPECTED CS ADDRESS
0016  7497 ;       RECEIVED CS ADDRESS
0016  7498 ;       LOOP COUNT I
0016  7499 ;
0016  7500 ;--
0016  7501 ;*******************************************************************************
0016  7502
0016  7503
0016  7504         INITIALIZE                                      ;INIT THE CPU
```

```
                                  0018          LOOP            I,1,2                   ;CREATE A TEST LOOP OF 2
                                  001F          LOADDCS         1$,I,02,1               ;LOAD MICRO CODE FOR THIS LOOP
                                  0026          ERRLOOP                                 ;ERROR LOOP POINT
                                  0028          FETCH           0                       ;START DCS PROGRAM
                                  002D          BRSTCLK         5                       ;END DCS PROGRAM
                                  0031          CMPREGMSK       CSTRP,2$,I,3$,,W,       ;TEST CS ADDRESS
                                  003D          IFERROR         ,<SPA>,                 ;INCORRECT BRANCH ADDRESS
                                  0043          ENDLOOP         I                       ;END TEST LOOP
                                  0046          SKIP                                    ;GO TO NEXT TEST
                                  004D
                                  004D
                                  004D  BEGIN_TEST_DATA
                                  004D
                                  004D      ;-----------------------------------------------------------------------
                                  004D
                                  004D  1$:
U 02, 7701,4815,0364,0200,0470,1803  004D  ;% 02   MSRC/PSHADD,SIZE[LONG]                 ;MICRO CODE LOOP 1
U 02, 7701,0814,0364,0200,0470,1803  0058  ;% 02   MSRC/PSHSUB,SIZE[LONG]                 ;MICRO CODE LOOP 2
                             1802  0063  2$:     .WORD   ^X1802                          ;RESULTS LOOP 1
                             1800  0065          .WORD   ^X1800                          ;RESULTS LOOP 2
                             3FFF  0067  3$:     .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                                  0069
                                  0069
                                  0069  BEGIN_CPU_DATA
                                  0002
                                  0002  DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 7700,8800,0364,03DC,0470,1802  000C  ;% 01   RSRC/ZERO.CLRRBSP                       ;CLEAR THE STACK POINTER
U 02, 7701,4815,0364,0200,0470,1803  0017  ;% 02   MSRC/PSHADD,SIZE[LONG]                 ;PUSH THE STACK
U 03, 7700,8800,0364,03DC,0470,1804  0022  ;% 03   RSRC/ZERO.CLRRBSP                       ;CLEAR THE STACK POINTER
U 04, 7700,081C,0364,BB00,0470,1800  002D  ;% 04   MSRC/READRBS,BUT/SPASTA,NEXT/1800       ;BRANCH ON RBS<6>
U 05, 7300,4800,0364,0300,0470,1806  0038  ;% 05   NOP,STOP.CLOCK                          ;STOP THE CPU CLOCK
                                  0043
                                  0043  END_CPU_DATA
                                  0069
                                  0069
                                  0069  END_TEST_DATA
                                  0069
                                  0069      ;-----------------------------------------------------------------------
                                  0069  ENDTEST
```

```
                        0016        .SBTTL  "TEST   0B2     BUT/SPASTA TEST 3
                        0016  7567 ;*******************************************************************************
                        0016  7568 ;++
                        0016  7569 ;
                        0016  7570 ; FUNCTIONAL DESCRIPTION:
                        0016  7571 ;
                        0016  7572 ;       THIS IS ONE IN A SERIES OF 4 TESTS THAT CHECK THE SPASTA BRANCHING
                        0016  7573 ;       LOGIC.
                        0016  7574 ;
                        0016  7575 ;
                        0016  7576 ; TEST ALGORITHM:
                        0016  7577 ;
                        0016  7578 ;       1.  CREATE A TEST LOOP I = 2
                        0016  7579 ;       2.  LOAD MICRO CODE INTO DCS ACCORDING TO I
                        0016  7580 ;       3.  EXECUTE DCS PROGRAM
                        0016  7581 ;               a.  ZERO THE RBSP (REGISTER BACKUP STACK POINTER)
                        0016  7582 ;               b.  READ THE RBS TO CAUSE THE RBSP TO INCREMENT (LOOP 2 ONLY)
                        0016  7583 ;               c.  BRANCH ON VALUE OF RBSP
                        0016  7584 ;               d.  STOP THE CPU CLOCK
                        0016  7585 ;       4.  TEST THE CS ADDRESS FOR THE CORRECT BRANCH
                        0016  7586 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING LOOP
                        0016  7587 ;
                        0016  7588 ;
                        0016  7589 ; TEST PATTERNS:
                        0016  7590 ;
                        0016  7591 ;       ITERATION       CS ADDRESS
                        0016  7592 ;           1               1801
                        0016  7593 ;           2               1800
                        0016  7594 ;
                        0016  7595 ;
                        0016  7596 ; LOGIC DESCRIPTION:
                        0016  7597 ;
                        0016  7598 ;       THIS TEST EXERCISES THE LOGIC INTERNAL TO THE SPA GATE ARRAY
                        0016  7599 ;       THAT IS RESPONSIBLE FOR THE SPA ST1 AND ST0 BITS DURING THE MICRO
                        0016  7600 ;       ORDER MSRC/WB_RBSP.  ANY FAILURES SHOULD BE CONFINED TO THE SPA
                        0016  7601 ;       GATE ARRAY OR THE DISCRETE BRANCHING LOGIC ON DPM.
                        0016  7602 ;
                        0016  7603 ;
                        0016  7604 ; ASSUMPTIONS:
                        0016  7605 ;
                        0016  7606 ;       THIS TEST ASSUMES THE WBUS IS OPERATIONAL.
                        0016  7607 ;
                        0016  7608 ;
                        0016  7609 ; ERROR DESCRIPTION:
                        0016  7610 ;
                        0016  7611 ;       EXPECTED CS ADDRESS
                        0016  7612 ;       RECEIVED CS ADDRESS
                        0016  7613 ;       LOOP COUNT I
                        0016  7614 ;
                        0016  7615 ;--
                        0016  7616 ;*******************************************************************************
                        0016  7617
                        0016  7618
                        0016  7619          INITIALIZE                              ;INIT THE CPU
                        0018  7620          LOOP            I,1,2                   ;CREATE A TEST LOOP OF 2
```

```
                                001F          LOADDCS        1$,I,02,1                ;MODIFY MICRO CODE FOR LOOP
                                0026          ERRLOOP                                 ;ERROR LOOP POINT
                                0028          FETCH          0                        ;START DCS PROGRAM
                                002D          BRSTCLK        4                        ;END DCS PROGRAM
                                0031          CMPREGMSK      CSTRP,2$,I,3$,,W,        ;TEST CS ADDRESS
                                003D          IFERROR        ,<SPA>,                  ;INCORRECT BRANCH ADDRESS
                                0043          ENDLOOP        I                        ;END TEST LOOP
                                0046          SKIP                                    ;GO TO NEXT TEST
                                004D
                                004D
                                004D  BEGIN_TEST_DATA
                                004D
                                004D      ;-----------------------------------------------------------------------
                                004D
                                004D  1$:
U 02, 7700,4800,0364,0300,0470,1803  004D  ;% 02   NOP                                    ;MICRO CODE LOOP 1
U 02, 7700,081C,0364,0300,0470,1803  0058  ;% 02   MSRC/READRBS                           ;MICRO CODE LOOP 2
                             1801  0063  2$:     .WORD   ^X1801                         ;RESULTS LOOP 1
                             1800  0065          .WORD   ^X1800                         ;RESULTS LOOP 2
                             3FFF  0067  3$:     .WORD   ^X3FFF                         ;MASK FOR CMPREGMSK PSEUDO OP
                                0069
                                0069
                                0069  BEGIN_CPU_DATA
                                0002
                                0002  DCS_DATA
                                0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 7700,8800,0364,03DC,0470,1802  000C  ;% 01   RSRC/ZERO.CLRRBSP                      ;ZERO THE RBSP
U 02, 7700,4800,0364,0300,0470,1803  0017  ;% 02   NOP
U 03, 7700,481E,0364,BB00,0470,1800  0022  ;% 03   BUT/SPASTA,MSRC/WB_RBSP,RSRC/TEMP0,NEXT/1800 ;BRANCH ON RBSP VALUE
U 04, 7300,4800,0364,0300,0470,1805  002D  ;% 04   NOP,STOP.CLOCK                         ;STOP THE CPU CLOCK
                                0038
                                0038  END_CPU_DATA
                                0069
                                0069
                                0069  END_TEST_DATA
                                0069
                                0069      ;-----------------------------------------------------------------------
                                0069  ENDTEST
```

```
                        0016        .SBTTL  "TEST   0B3     BUT/SPASTA TEST 4
                        0016  7678 ;*******************************************************************************
                        0016  7679 ;++
                        0016  7680 ;
                        0016  7681 ; FUNCTIONAL DESCRIPTION:
                        0016  7682 ;
                        0016  7683 ;       THIS TEST IS ONE IN A SERIES OF 4 TESTS THAT CHECK THE SPASTA BRANCHING
                        0016  7684 ;       LOGIC.
                        0016  7685 ;
                        0016  7686 ;
                        0016  7687 ; TEST ALGORITHM:
                        0016  7688 ;
                        0016  7689 ;       1.  CREATE A TEST LOOP I = 4
                        0016  7690 ;       2.  PUT A TEST PATTERN INTO THE WDREG
                        0016  7691 ;       3.  EXECUTE DCS PROGRAM
                        0016  7692 ;               a.  PUT THE TEST PATTERN INTO THE RNUM REGISTER
                        0016  7693 ;               b.  BRANCH THE CS ADDRESS ACCORDING TO RNUM VALUE
                        0016  7694 ;               c.  STOP THE CPU CLOCK
                        0016  7695 ;       4.  TEST THE CS ADDRESS FOR THE CORRECT BRANCH
                        0016  7696 ;       5.  IF NO ERROR GO TO STEP 2 UNTIL THE TEST LOOP IS EXHAUSTED
                        0016  7697 ;
                        0016  7698 ;
                        0016  7699 ; TEST PATTERNS:
                        0016  7700 ;
                        0016  7701 ;       ITERATION       TEST PATTERN    CS ADDRESS
                        0016  7702 ;           1             00000000        1800
                        0016  7703 ;           2             00000006        1803
                        0016  7704 ;           3             00000007        1802
                        0016  7705 ;           4             0000000E        1801
                        0016  7706 ;
                        0016  7707 ;
                        0016  7708 ; LOGIC DESCRIPTION:
                        0016  7709 ;
                        0016  7710 ;       THIS TEST EXERCISES THE LOGIC INTERNAL TO THE SPA GATE ARRAY THAT IS
                        0016  7711 ;       RESPONSIBLE FOR THE SPA ST1 AND ST0 BITS.  ANY FAILURE SHOULD BE
                        0016  7712 ;       CONFINED TO THE SPA GATE ARRAY OR THE DISCRETE BRANCHING LOGIC ON
                        0016  7713 ;       DPM.
                        0016  7714 ;
                        0016  7715 ;
                        0016  7716 ; ASSUMPTIONS:
                        0016  7717 ;
                        0016  7718 ;       THIS TEST ASSUMES THE WBUS IS OPERATIONAL.
                        0016  7719 ;
                        0016  7720 ;
                        0016  7721 ; ERROR DESCRIPTION:
                        0016  7722 ;
                        0016  7723 ;       EXPECTED CS ADDRESS
                        0016  7724 ;       RECEIVED CS ADDRESS
                        0016  7725 ;       LOOP COUNT I
                        0016  7726 ;
                        0016  7727 ;--
                        0016  7728 ;*******************************************************************************
                        0016  7729
                        0016  7730
                        0016  7731         INITIALIZE                                  ;INIT THE CPU
```

```
                                  0018  7732          LOOP            I,1,4                   ;CREATE A TEST LOOP OF 4
                                  001F  7733          LOADREG         WDREG,1$,I,L            ;PUT TEST PATTERN INTO WDREG
                                  0027  7734          ERRLOOP                                 ;ERROR LOOP POINT
                                  0029  7735          FETCH           0                       ;START DCS PROGRAM
                                  002E  7736          BRSTCLK         3                       ;END DCS PROGRAM
                                  0032  7737          CMPREGMSK       CSTRP,2$,I,3$,,W,       ;TEST CS ADDRESS
                                  003E  7738          IFERROR         ,<SPA>,                 ;INCORRECT ADDRESS
                                  0044  7739          ENDLOOP         I                       ;END TEST LOOP
                                  0047  7740          SKIP                                    ;GO TO NEXT TEST
                                  004E  7741
                                  004E  7742
                                  004E  7743 BEGIN_TEST_DATA
                                  004E
                                  004E        ;-----------------------------------------------------------------------
                                  004E  7744
                         00000000 004E  7745 1$:      .LONG   ^X00000000                      ;TEST PATTERNS
                         00000006 0052  7746          .LONG   ^X00000006
                         00000007 0056  7747          .LONG   ^X00000007
                         0000000E 005A  7748          .LONG   ^X0000000E
                             1800 005E  7749 2$:      .WORD   ^X1800                          ;RESULTS TABLE
                             1803 0060  7750          .WORD   ^X1803
                             1802 0062  7751          .WORD   ^X1802
                             1801 0064  7752          .WORD   ^X1801
                             3FFF 0066  7753 3$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                                  0068  7754
                                  0068  7755
                                  0068  7756 BEGIN_CPU_DATA
                                  0002  7757
                                  0002  7758 DCS_DATA
                                  0001  7759
U 00, 7700,C800,0364,0300,0470,1801 0001  7760 ;% 00    NOP
U 01, 5700,C81D,0364,0300,0470,1802 000C  7764 ;% 01    MSRC/RNUM_WBUS,WB_RDM                   ;LOAD RNUM REGISTER
U 02, 7700,0800,0364,BB40,0470,1800 0017  7768 ;% 02    BUT/SPASTA,RSRC/R0,NEXT/1800            ;BRANCH ON RNUM VALUE
U 03, 7300,C800,0364,0300,0470,1804 0022  7772 ;% 03    NOP,STOP.CLOCK                          ;STOP THE CPU CLOCK
                                  002D  7776
                                  002D  7777 END_CPU_DATA
                                  0068  7778
                                  0068  7779
                                  0068  7780 END_TEST_DATA
                                  0068
                                  0068        ;-----------------------------------------------------------------------
                                  0068  7781 ENDTEST
```

```
                        001E        .SBTTL  "TEST   0B4     PRINT MICRO CODE REVISION
                        001E  7783 ;*******************************************************************************
                        001E  7784 ;++
                        001E  7785 ;
                        001E  7786 ; FUNCTIONAL DESCRIPTION:
                        001E  7787 ;
                        001E  7788 ;       THIS TEST WILL READ AND PRINT THE MICRO CODE REVISION LEVEL.
                        001E  7789 ;
                        001E  7790 ;
                        001E  7791 ; TEST ALGORITHM:
                        001E  7792 ;
                        001E  7793 ;       1.  EXECUTE DCS PROGRAM
                        001E  7794 ;               a.  ZERO THE D REGISTER
                        001E  7795 ;       2.  CLEAR RDM MASTER HALT ENABLE TO FREE UP THE CLOCKS
                        001E  7796 ;       3.  SET MICRO ADDRESS INHIBIT TO GET EVERYBODY OFF THE CS ADDRESS BUS
                        001E  7797 ;       4.  LOAD ADDRESS FF6 INTO RDM CSAMR REGISTER
                        001E  7798 ;       5.  ENABLE FF6 ONTO CS ADDRESS BUS
                        001E  7799 ;       6.  STEP THE CPU CLOCK TO EXECUTE THE MICRO INSTRUCTION AT FF6
                        001E  7800 ;       7.  EXEXUTE DCS PROGRAM
                        001E  7801 ;               a.  PUT THE MICRO CODE REVISION LEVEL IN RDM WHREG
                        001E  7802 ;       8.  PRINT THE REVISION LEVEL
                        001E  7803 ;
                        001E  7804 ;
                        001E  7805 ; TEST PATTERNS:
                        001E  7806 ;
                        001E  7807 ;       NONE
                        001E  7808 ;
                        001E  7809 ;
                        001E  7810 ; LOGIC DESCRIPTION:
                        001E  7811 ;
                        001E  7812 ;       THIS TEST IS STRICTLY FOR CONVENIENCE TO THE USER.  THE MICRO CODE
                        001E  7813 ;       REVISION LEVEL IS READ FROM THE CPU CONTROL STORE AND PRINTED
                        001E  7814 ;       ON THE CONSOLE.
                        001E  7815 ;
                        001E  7816 ;
                        001E  7817 ; ASSUMPTIONS:
                        001E  7818 ;
                        001E  7819 ;       THIS TEST ASSUMES THAT THE CPU CONTROL STORE IS OPERATIONAL.
                        001E  7820 ;
                        001E  7821 ;
                        001E  7822 ; ERROR DESCRIPTION:
                        001E  7823 ;
                        001E  7824 ;       MICRO CODE REVISION LEVEL IS *
                        001E  7825 ;
                        001E  7826 ;--
                        001E  7827 ;*******************************************************************************
                        001E  7828
                        001E  7829
                        001E  7830         INITIALIZE                              ;INIT THE CPU
                        0020  7831         FETCH           0                       ;START DCS PROGRAM
                        0025  7832         BRSTCLK         2                       ;END DCS PROGRAM
                        0029  7833         LOADREG         RDCTRL,8$.,B            ;CLEAR MASTER HALT ENABLE
                        0031  7834         LOADREG         DCSCR,1$.,B             ;SET UADDR INH
                        0039  7835         LOADREG         CSAMR,2$.,W             ;LOAD ADDRESS FF6
                        0041  7836         LOADREG         DCSCR,3$.,B             ;SET PARCHK
```

```
                        0049  7837          LOADREG         DCSCR,4$,,B             ;SINGLE STEP THE CLOCK
                        0051  7838          LOADREG         DCSCR,1$,,B             ;STOP THE CLOCK
                        0059  7839          LOADREG         DCSCR,4$,,B             ;SINGLE STEP THE CLOCK
                        0061  7840          LOADREG         DCSCR,5$,,B             ;CLEAR UADDR INH
                        0069  7841          FETCH           3                       ;START DCS PROGRAM
                        006E  7842          BRSTCLK         5                       ;END DCS PROGRAM
                        0072  7843          PRINT_S         6$                      ;PRINT THE REV LEVEL
                        0076  7844          PRINT_N         WHREG,B                 ;...
                        007B  7845          PRINT_S         7$                      ;...
                        007F  7846          SKIP                                    ;GO TO THE NEXT TEST
                        0086  7847
                        0086  7848
                        0086  7849 BEGIN_TEST_DATA
                        0086
                        0086              ;-------------------------------------------------------------------------
                        0086  7850
                     E0 0086  7851 1$:      .BYTE   ^XE0                            ;SET UADDR INH
                   0FF6 0087  7852 2$:      .WORD   ^X0FF6                          ;ADDRESS OF MICRO REV LEVEL
                     E8 0089  7853 3$:      .BYTE   ^XE8                            ;SET PARCHK
                     A8 008A  7854 4$:      .BYTE   ^XA8                            ;SINGLE STEP THE CLOCK
                     60 008B  7855 5$:      .BYTE   ^X60                            ;CLEAR UADDR INH
44 4F 43 20 4F 52 43 49 4D 0A 0D 00' 008C  7856 6$:      .ASCIC  <13><10>/MICRO CODE REVISION LEVEL IS /
4C 20 4E 4F 49 53 49 56 45 52 20 45  0098
          20 53 49 20 4C 45 56 45    00A4
                                1F   008C
                        0A 0D 00' 00AC  7857 7$:      .ASCIC  <13><10>
                                02   00AC
                                00   00AF  7858 8$:      .BYTE   ^X00
                                     00B0  7859
                                     00B0  7860 BEGIN_CPU_DATA
                                     0002  7861
                                     0002  7862 DCS_DATA
                                     0001  7863
U 00, 7700,C800,0364,0300,0470,1801  0001  7864 ;% 00    NOP
U 01, 7700,8800,5BE6,03D8,0470,1802  000C  7868 ;% 01    D_R[ZERO]                          ;ZERO THE D REGISTER
U 02, 7300,4800,0364,0300,0470,1803  0017  7872 ;% 02    NOP,STOP.CLOCK                     ;STOP THE CPU CLOCK
U 03, 7700,C800,0364,0300,0470,1804  0022  7876 ;% 03    NOP
U 04, 6701,0800,2B70,0008,0470,1805  002D  7880 ;% 04    RDM_WB,WB_R[TEMP2].RR.8            ;MOVE MICRO CODE REV TO WHREG
U 05, 7300,4800,0364,0300,0470,1806  0038  7884 ;% 05    NOP,STOP.CLOCK                     ;STOP THE CPU CLOCK
                                     0043  7888
                                     0043  7889 END_CPU_DATA
                                     00B0  7890
                                     00B0  7891
                                     00B0  7892 END_TEST_DATA
                                     00B0
                                     00B0              ;-------------------------------------------------------------------------
                                     00B0  7893 ENDTEST
```

```
                        0022        .SBTTL  "TEST   0B5     PRINT HARDWARE REVISION LEVEL
                        0022  7895 ;*******************************************************************************
                        0022  7896 ;++
                        0022  7897 ;
                        0022  7898 ; FUNCTIONAL DESCRIPTION:
                        0022  7899 ;
                        0022  7900 ;       THIS TEST READS AND PRINTS THE CONTENTS OF THE HARDWARE REVISION LEVEL
                        0022  7901 ;       JUMPERS.
                        0022  7902 ;
                        0022  7903 ;
                        0022  7904 ; TEST ALGORITHM:
                        0022  7905 ;
                        0022  7906 ;       1.  EXECUTE DCS PROGRAM
                        0022  7907 ;               a.  READ HARDWARE REVISION LEVEL TO RTEMP0
                        0022  7908 ;               b.  ROTATE REVISION LEVEL TO LOW BYTE AND READ INTO RDM WHREG
                        0022  7909 ;       2.  PRINT THE REVISION LEVEL
                        0022  7910 ;
                        0022  7911 ;
                        0022  7912 ; TEST PATTERNS:
                        0022  7913 ;
                        0022  7914 ;       NONE
                        0022  7915 ;
                        0022  7916 ;
                        0022  7917 ; LOGIC DESCRIPTION:
                        0022  7918 ;
                        0022  7919 ;       THIS TEST IS STRICTLY FOR THE CONVENIENCE OF THE USER.  IT PREVENTS YOU
                        0022  7920 ;       FROM HAVING TO TRY AND FIGUREOUT WHAT THE JUMPERS ON THE BACKPLANE ARE
                        0022  7921 ;       SET TO.
                        0022  7922 ;
                        0022  7923 ;
                        0022  7924 ; ASSUMPTIONS:
                        0022  7925 ;
                        0022  7926 ;       THIS TEST ASSUMES THE WBUS AND ROTATOR ARE OPERATIONAL.
                        0022  7927 ;
                        0022  7928 ;
                        0022  7929 ; ERROR DESCRIPTION:
                        0022  7930 ;
                        0022  7931 ;       HARDWARE REVISION LEVEL IS *
                        0022  7932 ;
                        0022  7933 ;--
                        0022  7934 ;*******************************************************************************
                        0022  7935
                        0022  7936
                        0022  7937
                        0022  7938         INITIALIZE                              ;INIT THE CPU
                        0024  7939         FETCH           0                       ;START DCS PROGRAM
                        0029  7940         BRSTCLK         3                       ;END DCS PROGRAM
                        002D  7941         PRINT_S         1$                      ;PRINT HARDWARE REV LEVEL
                        0031  7942         PRINT_N         WHREG,B                 ;...
                        0036  7943         PRINT_S         2$                      ;...
                        003A  7944         SKIP                                    ;GO TO NEXT TEST
                        0041  7945
                        0041  7946
                        0041  7947 BEGIN_TEST_DATA
                        0041
```

```
                                        0041        ;-----------------------------------------------------------------------
                                        0041  7948
20 45 52 41 57 44 52 41 48 0A 0D 00'    0041  7949 1$:      .ASCIC  <13><10>/HARDWARE REVISION LEVEL IS /
56 45 4C 20 4E 4F 49 53 49 56 45 52     004D
                  20 53 49 20 4C 45     0059
                                  1D    0041
                           0A 0D 00'    005F  7950 2$:      .ASCIC  <13><10>
                                  02    005F
                                        0062  7951
                                        0062  7952
                                        0062  7953
                                        0062  7954 BEGIN_CPU_DATA
                                        0002  7955
                                        0002  7956 DCS_DATA
                                        0001  7957
U 00, 7700,C800,0364,0300,0470,1801     0001  7958 ;% 00    NOP
U 01, 7700,0840,0364,0300,2270,1802     000C  7962 ;% 01    WCTRL/REVLEVEL,R[TEMP0]_WB                ;READ HARDWARE REV LEVEL
U 02, 6701,0800,2B70,0100,0470,1803     0017  7966 ;% 02    RDM_WB,WB_R[TEMP0].RR.16                  ;ROTATE TO LOW BYTE
U 03, 7300,C800,0364,0300,0470,1804     0022  7970 ;% 03    NOP,STOP.CLOCK                            ;STOP THE CPU CLOCK
                                        002D  7974
                                        002D  7975 END_CPU_DATA
                                        0062  7976
                                        0062  7977
                                        0062  7978 END_TEST_DATA
                                        0062
                                        0062        ;-----------------------------------------------------------------------
                                        0062  7979 ENDTEST
```

```
                0000  7981 .END
```

```
$CONTROL_C_CODE= 0000000A
$FILE_NOT_FOUND= 00000009
$STN            = 00000000
$TEST_NUMBER    = 000000B6
ACV             = 0000000F
AC_LOW          = 00000080
ADD             = 00000019
ADDR_MATCH_HI   = 00007405
ADDR_MATCH_LO   = 00007404
ADK             = 0000000B
ALK             = 00000000
ALP             = 00000016
ARG_CNT         = 0000003F R     D7
A_REG_BYTE_0    = 00007410
A_REG_BYTE_1    = 00007411
A_REG_BYTE_2    = 00007412
A_REG_BYTE_3    = 00007413
B               = 00000001
BADD1           = 00007426
BELL_FLAG       = 00000008
BELL_KEY        = 0000000E
BUFF_ADDR_LOW   = 00007426
BURST_STOP_FLAG= 00000080
BYTE            = 00000001
CACHE_FLAG      = 00000000
CACHE_FLAG1     = 00000000
CACHE_HEAD      = 00000000 R     9E
CACHE_LENGTH    = 00000001
CAK             = 0000000A
CCC             = 00000004
CCS             = 00000005
CF_KEY          = 00000024
CLA             = 00000003
CLEAR_CTRL_FILE= 00000002
CLKDCS          = 00007427
CLK_CTRL_0      = 00000020
CLK_CTRL_0_RO   = 00000040
CLK_CTRL_1      = 00000040
CLK_CTRL_1_RO   = 00000080
CLOCK_STOPED    = 00000080
CMI             = 00000007
CMICTL          = 0000741C
CMI_COMPLETE    = 00000020
CMI_CTRL_CLEAR  = 00000001
CMI_CTRL_REG    = 0000741C
CMI_GO          = 00000008
CMI_READ        = 00000040
CMI_STATUS_0    = 00000008
CMI_STATUS_1    = 00000010
CMI_WRITE       = 00000020
CML             = 0000000E
COMPOSIT_LENGTH= 00000002
CON             = 00000011
CONTINUE_FLAG   = 00000002
CONTROL_C_FLAG  = 00000001
CONT_FLAG       = 00000020
CONT_KEY        = 0000001C
CPU_RUN         = 00000010
CSAMR           = 00007404
CSBUF           = 00007424
CSTRP           = 00007422
CS_ADD_BUFF_HGH= 00007425
CS_ADD_BUFF_LOW= 00007424
CS_ADD_TRAP_HGH= 00007423
CS_ADD_TRAP_LOW= 00007422
CYCLE_FLAG      = 00000008
CYCLE_KEY       = 00000020
D0              = 00000005
D1              = 0000000B
D2              = 00000000
DATA_FLAG       = 00000000
DATA_LENGTH     = 00000001
DCSAD           = 00007401
DCSCF           = 00007403
DCSCR           = 00007402
DCSDA           = 00007400
DCS_ADDR_CLEAR  = 00000002
DCS_ADDR_REG_RO= 00007427
DCS_ADDR_REG_WO= 00007401
DCS_CTRL_FILE   = 00007403
DCS_CTRL_REG    = 00007402
DCS_DATA_REG    = 00007400
DCS_FLAG        = 00000000
DCS_FLAG1       = 00000000
DCS_HEAD        = 00000000 R     E4
DCS_LENGTH      = 00000001
DCS_SCRATCH_ADR= 00000036
DCS_START       = 00001800
DC_LOW          = 00000040
DIAGNOSE_FLAG   = 00000004
DPM             = 00000000
DUM1            = FFFFFFF5
ENDTEST_LOC     = 00000067 R     E3
END_SLICE       = 00000025
END_TEST_143      000001A2 R     02
END_TEST_144      000000F0 R     09
END_TEST_145      000001E1 R     0F
END_TEST_146      000000C4 R     15
END_TEST_147      0000012C R     1B
END_TEST_148      000000D2 R     21
END_TEST_149      00000155 R     27
END_TEST_150      00000128 R     2D
END_TEST_151      00000186 R     33
END_TEST_152      00000105 R     39
END_TEST_153      000000CA R     3F
END_TEST_154      000000E0 R     45
END_TEST_155      00000097 R     4B
END_TEST_156      000000AC R     51
END_TEST_157      00000163 R     57
END_TEST_158      00000105 R     5D
END_TEST_159      00000088 R     63
END_TEST_160      000000DA R     69
END_TEST_161      00000455 R     6F
END_TEST_162      000002A9 R     75
END_TEST_163      00000125 R     77
END_TEST_164      00000145 R     7D
END_TEST_165      000001E1 R     83
END_TEST_166      00000085 R     89
END_TEST_167      0000012A R     8F
END_TEST_168      000000CB R     95
END_TEST_169      0000006A R     9B
END_TEST_170      00000172 R     A1
END_TEST_171      000000ED R     A7
END_TEST_172      000000B4 R     AD
END_TEST_173      000000DF R     B3
END_TEST_174      000000D4 R     B9
END_TEST_175      00000123 R     BF
END_TEST_176      00000079 R     C5
END_TEST_177      00000069 R     CB
END_TEST_178      00000069 R     D1
END_TEST_179      00000068 R     D7
END_TEST_180      000000B0 R     DD
END_TEST_181      00000062 R     E3
EN_TRACE_REG    = 00000008
ERRLOG_FLAG     = 00000000
ERROR_FLAG      = 00000010
ERROR_LOG_ADR   = 000032FC
ERR_LOG_INIT    = 00000001
EXP_STALL_FLAG  = 00000010
EXP_UTRAP_FLAG  = 00000040
FAC             = 00000020
FAULT           = 00000002
FCC             = 00000015
FCS             = 00000021
FETCH_FLAG      = 00000008
FEX             = 0000001E
FFH             = 00000024
FFL             = 00000023
FIC             = 0000001F
FIO             = 0000001D
FLAG_KEY        = 00000004
FMR             = 00000022
FPA             = 00000002
FP_BOOT_0       = 00000001
FP_BOOT_1       = 00000002
FP_START_0      = 00000004
FP_START_1      = 00000008
FQA             = 00000014
FRONT1          = 00007420
FRONT2          = 00007421
FRONT_PNL_1     = 00007420
FRONT_PNL_2     = 00007421
FRONT_PNL_LOCK  = 00000080
HALT_FLAG       = 00000001
HALT_KEY        = 00000008
```

```
HALT_ON_MATCH   = 00000001          MD_REG_BYTE_0   = 00007414          PAR_STOP_ENABL  = 00000010
HEAD            = FFFF7C00          MD_REG_BYTE_1   = 00007415          PASS_KEY        = 00000002
HIGH            = 00000001          MD_REG_BYTE_2   = 00007416          PB_INIT         = 00000040
H_FROM_WBUS     = 00000010          MD_REG_BYTE_3   = 00007417          PB_KEY          = 00000036
IB_FLAG         = 00000020          MEC             = 0000001A          PC_KEY          = 00000030
IB_KEY          = 00000012          MHREG           = 00007418          PHB             = 00000007
INSTR_FLAG      = 00000004          MH_REG_BYTE_0   = 00007418          PRK             = 0000000C
INSTR_KEY       = 00000018          MH_REG_BYTE_1   = 00007419          PS_KEY          = 0000003E
INT             = 00000010          MH_REG_BYTE_2   = 0000741A          QA_FLAG         = 00000040
IRD             = 00000005          MH_REG_BYTE_3   = 0000741B          QA_KEY          = 00000014
I_INDEX_FLAG    = 00000000          MIC             = 00000001          QV_FLAG         = 00000002
I_INIT          = 00000000          MICROWORD       = 0000000A          QV_KEY          = 00000028
J_INDEX_FLAG    = 00000000          MICRO_ADDR_INH  = 00000080          RDCTRL          = 0000741E
J_INIT          = 00000000          MONITOR_SIZE    = 00002C5E          RDM             = 00000004
K_INDEX_FLAG    = 00000000          MSQ             = 00000008          RDM_FLAG        = 00000004
K_INIT          = 00000000          MT_KEY          = 0000002E          RDM_KEY         = 0000002A
L               = 00000004          N               = 00000001          RDM_LAST        = 00000008
LIST_END        = 00000044 R    D7  NER_FLAG        = 00000004          RD_CTRL_REG     = 0000741E
LIST_HEAD       = 00000040 R    D7  NER_KEY         = 0000000C          REMOTE          = 00000001
LIST_LENGTH     = 00000004          NOERROR         = 00000001          RTEMP_FLAG      = 00000000
LITERAL         = 00000000          NOTEQUAL        = 00000003          RTEMP_FLAG1     = 00000000
LONG            = 00000004          ONEQUAL         = 00000002          RTEMP_HEAD      = 00000000 R    C1
LOOP_ERROR_FLAG= 00000020           ONERROR         = 00000000          RTEMP_LENGTH    = 00000001
LOOP_FLAG       = 00000002          OP_BEGIN_LOOP   = 00000008          RT_KEY          = 0000002C
LOOP_KEY        = 0000000A          OP_BURST_CLOCK  = 00000002          SAC             = 00000009
LOST_FLAG       = 00000010          OP_COMPARE_REG  = 00000004          SA_CLOCK        = 00000080
LOST_KEY        = 0000001A          OP_COMPARE_REGM= 00000006           SA_FLAG         = 00000010
LOW             = 00000000          OP_COMPARE_VB   = 00000020          SA_KEY          = 00000010
M               = 0000000A          OP_DUMPLOG      = 00000030          SA_START_STOP   = 00000080
M$TEST_SIZE     = 000007A2          OP_ENABL_STALL  = 00000038          SBE_FLAG        = 00000020
M$TEST_SIZE_D   = 000003E2          OP_ENABL_TRAP   = 0000002E          SBE_KEY         = 00000042
MA0             = 00000008          OP_END_FILE     = 00000024          SECTION_FLAG    = 00000080
MA1             = 0000000A          OP_END_LOOP     = 0000000A          SF_KEY          = 00000040
MAD             = 00000013          OP_END_TEST     = 0000000C          SOMM_FLAG       = 00000001
MAIN32          = 0000741D          OP_ERRLOG       = 00000032          SOMM_KEY        = 00000006
MAINT_A_TO_CMI  = 00000080          OP_ERROR_LOOP   = 0000000E          SPA             = 00000002
MAINT_CMI_TO_MH= 00000020           OP_FETCH        = 00000022          SRK             = 00000001
MAINT_DCS_ENABL= 00000004           OP_IF_ERROR     = 00000012          SRM             = 00000017
MAINT_MD_TO_CMI= 00000040           OP_INC_INDEX    = 0000002A          SR_KEY          = 0000003C
MAINT_REG       = 0000741D          OP_INIT         = 00000014          STACK_SIZE      = 00000400
MAINT_STB_INH   = 00000002          OP_LOAD_DCS     = 00000000          STATUS          = 00007406
MAINT_TRAP_OFF  = 00000001          OP_LOAD_FIELD   = 00000026          STATUS_REG      = 00007406
MAINT_WB_TO_WH  = 00000008          OP_LOAD_INDEX   = 00000028          STEP_KEY        = 0000001E
MAINT_WD_TO_WB  = 00000010          OP_LOAD_REG     = 00000016          STOP_CLOCK      = 00000004
MAP             = 00000012          OP_MASK         = 00000018          STROBE_CMI      = 00000040
MAREG           = 00007410          OP_NEW_TEST     = 0000001A          ST_KEY          = 0000001E
MASTER_HALT_EN  = 00000008          OP_PATTERN_GEN  = 00000010          TEMP            = 00000002 R    E2
MAX_ARGUMENTS   = 00000010          OP_PRINT_N      = 00000036          TEMP1           = 0000004F
MA_KEY          = 00000038          OP_PRINT_S      = 00000034          TEMP2           = 0000002F
MC0             = 00000003          OP_SAVE_INDEX   = 0000002C          TEMP3           = 00000002
MC1             = 00000009          OP_SKIP         = 0000001C          TEMP4           = 00000002 R    E2
MDL             = 0000001B          OP_SUB_TEST     = 0000001E          TEST            = 00000004
MDR             = 00000018          OVERLAY_HEAD    = 00000000 R    E8  TEST_FLAG       = 00000000
MDREG           = 00007414          PARITY_ERROR    = 00000020          TEST_HEAD       = 0000001F R    E3
MD_KEY          = 00000034          PAR_CHK_ENABL   = 00000008          TEST_KEY        = 00000000
```

```
TEST_LENGTH     = 00000001
TEST_SECT_HEAD  = 00000000 R      E3
THIS_TEST       = 000000B5
TICK_FLAG       = 00000002
TICK_KEY        = 00000022
TOK             = 00000006
TRACE_ENABL     = 00000004
TRAP_HALT_EN    = 00000020
TRAP_OFF        = 00000002
TR_FLAG         = 00000080
TR_KEY          = 00000016
TSTSPAN_FLAG    = 00000040
UBI             = 00000006
UDP             = 0000001C
UTR             = 0000000D
VA_KEY          = 00000032
VBUS_CLOCK      = 00000080
VBUS_COMPARE    = 00000080
VBUS_LOAD       = 00000010
VBUS_SERIAL_IN  = 00000040
VBUS_SERIAL_OUT = 00000001
VB_KEY          = 00000026
V_BUS_STROBE    = 00000001
W               = 00000002
WBUS_FROM_D     = 00000020
WDREG           = 0000742C
WD_KEY          = 0000003A
WD_REG_BYTE_0   = 0000742C
WD_REG_BYTE_1   = 0000742D
WD_REG_BYTE_2   = 0000742E
WD_REG_BYTE_3   = 0000742F
WHREG           = 00007430
WH_REG_BYTE_0   = 00007430
WH_REG_BYTE_1   = 00007431
WH_REG_BYTE_2   = 00007432
WH_REG_BYTE_3   = 00007433
WORD            = 00000002
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
X_0143_D	00000002 (2.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0143_T	00000200 (512.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0143_GDCS	00000090 (144.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0143_ERTEMP	0000002D (45.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0143_FCACHE	00000095 (149.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0143_HFILL	0000002C (44.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
DIRECTORY	00000027 (39.)	07 (7.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0144_D	00000002 (2.)	08 (8.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0144_T	00000100 (256.)	09 (9.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0144_GDCS	0000009B (155.)	0A (10.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0144_ERTEMP	00000031 (49.)	0B (11.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0144_FCACHE	00000029 (41.)	0C (12.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0144_HFILL	00000009 (9.)	0D (13.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0145_D	00000002 (2.)	0E (14.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0145_T	00000200 (512.)	0F (15.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0145_GDCS	00000090 (144.)	10 (16.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0145_ERTEMP	00000045 (69.)	11 (17.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0145_FCACHE	000000B5 (181.)	12 (18.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0145_HFILL	00000074 (116.)	13 (19.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0146_D	00000002 (2.)	14 (20.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0146_T	00000100 (256.)	15 (21.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0146_GDCS	0000009B (155.)	16 (22.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0146_ERTEMP	00000015 (21.)	17 (23.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0146_FCACHE	00000015 (21.)	18 (24.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0146_HFILL	00000039 (57.)	19 (25.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0147_D	00000002 (2.)	1A (26.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0147_T	00000180 (384.)	1B (27.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0147_GDCS	000000A6 (166.)	1C (28.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0147_ERTEMP	00000035 (53.)	1D (29.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0147_FCACHE	00000071 (113.)	1E (30.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0147_HFILL	00000032 (50.)	1F (31.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0148_D	00000002 (2.)	20 (32.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0148_T	00000100 (256.)	21 (33.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0148_GDCS	00000090 (144.)	22 (34.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0148_ERTEMP	00000039 (57.)	23 (35.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0148_FCACHE	00000035 (53.)	24 (36.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0148_HFILL	00000080 (128.)	25 (37.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0149_D	00000002 (2.)	26 (38.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0149_T	00000180 (384.)	27 (39.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0149_GDCS	00000090 (144.)	28 (40.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0149_ERTEMP	00000045 (69.)	29 (41.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0149_FCACHE	00000069 (105.)	2A (42.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0149_HFILL	00000040 (64.)	2B (43.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0150_D	00000002 (2.)	2C (44.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0150_T	00000180 (384.)	2D (45.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0150_GDCS	0000009B (155.)	2E (46.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0150_ERTEMP	00000059 (89.)	2F (47.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

X_0150_FCACHE	00000049	(73.)	30	(48.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0150_HFILL	00000041	(65.)	31	(49.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0151_D	00000002	(2.)	32	(50.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0151_T	00000200	(512.)	33	(51.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0151_GDCS	00000090	(144.)	34	(52.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0151_ERTEMP	00000021	(33.)	35	(53.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0151_FCACHE	00000085	(133.)	36	(54.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0151_HFILL	00000048	(72.)	37	(55.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0152_D	00000002	(2.)	38	(56.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0152_T	00000180	(384.)	39	(57.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0152_GDCS	0000009B	(155.)	3A	(58.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0152_ERTEMP	00000029	(41.)	3B	(59.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0152_FCACHE	00000035	(53.)	3C	(60.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0152_HFILL	00000005	(5.)	3D	(61.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0153_D	00000002	(2.)	3E	(62.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0153_T	00000100	(256.)	3F	(63.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0153_GDCS	00000085	(133.)	40	(64.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0153_ERTEMP	00000031	(49.)	41	(65.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0153_FCACHE	00000065	(101.)	42	(66.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0153_HFILL	00000063	(99.)	43	(67.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0154_D	00000002	(2.)	44	(68.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0154_T	00000100	(256.)	45	(69.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0154_GDCS	00000085	(133.)	46	(70.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0154_ERTEMP	00000041	(65.)	47	(71.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0154_FCACHE	00000079	(121.)	48	(72.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0154_HFILL	0000003F	(63.)	49	(73.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0155_D	00000002	(2.)	4A	(74.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0155_T	00000100	(256.)	4B	(75.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0155_GDCS	00000090	(144.)	4C	(76.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0155_ERTEMP	00000011	(17.)	4D	(77.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0155_FCACHE	00000055	(85.)	4E	(78.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0155_HFILL	00000008	(8.)	4F	(79.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0156_D	00000002	(2.)	50	(80.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0156_T	00000100	(256.)	51	(81.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0156_GDCS	000000B1	(177.)	52	(82.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0156_ERTEMP	00000011	(17.)	53	(83.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0156_FCACHE	00000039	(57.)	54	(84.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0156_HFILL	00000003	(3.)	55	(85.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0157_D	00000002	(2.)	56	(86.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0157_T	00000180	(384.)	57	(87.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0157_GDCS	00000038	(56.)	58	(88.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0157_ERTEMP	0000001D	(29.)	59	(89.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0157_FCACHE	00000001	(1.)	5A	(90.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0157_HFILL	00000028	(40.)	5B	(91.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0158_D	00000002	(2.)	5C	(92.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0158_T	00000180	(384.)	5D	(93.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0158_ERTEMP	00000009	(9.)	5E	(94.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0158_FCACHE	00000001	(1.)	5F	(95.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0158_GDCS	00000001	(1.)	60	(96.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0158_HFILL	00000073	(115.)	61	(97.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0159_D	00000002	(2.)	62	(98.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0159_T	00000100	(256.)	63	(99.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0159_GDCS	00000085	(133.)	64	(100.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0159_ERTEMP	00000015	(21.)	65	(101.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0159_FCACHE	00000011	(17.)	66	(102.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE

X_0159_HFILL	00000053	(83.)	67	(103.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0160_D	00000002	(2.)	68	(104.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0160_T	00000100	(256.)	69	(105.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0160_GDCS	0000002D	(45.)	6A	(106.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0160_ERTEMP	00000001	(1.)	6B	(107.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0160_FCACHE	00000001	(1.)	6C	(108.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0160_HFILL	0000004F	(79.)	6D	(109.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0161_D	00000002	(2.)	6E	(110.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0161_T	00000480	(1152.)	6F	(111.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0161_GDCS	00000064	(100.)	70	(112.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0161_ERTEMP	0000000D	(13.)	71	(113.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0161_FCACHE	00000001	(1.)	72	(114.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0161_HFILL	0000000C	(12.)	73	(115.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0162_D	00000002	(2.)	74	(116.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0162_T	000002FE	(766.)	75	(117.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0163_D	00000002	(2.)	76	(118.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0163_T	00000180	(384.)	77	(119.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0163_GDCS	00000022	(34.)	78	(120.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0163_ERTEMP	00000005	(5.)	79	(121.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0163_FCACHE	00000001	(1.)	7A	(122.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0163_HFILL	00000056	(86.)	7B	(123.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0164_D	00000002	(2.)	7C	(124.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0164_T	00000180	(384.)	7D	(125.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0164_GDCS	00000022	(34.)	7E	(126.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0164_ERTEMP	00000005	(5.)	7F	(127.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0164_FCACHE	00000001	(1.)	80	(128.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0164_HFILL	00000056	(86.)	81	(129.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0165_D	00000002	(2.)	82	(130.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0165_T	00000200	(512.)	83	(131.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0165_GDCS	00000022	(34.)	84	(132.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0165_ERTEMP	00000005	(5.)	85	(133.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0165_FCACHE	00000001	(1.)	86	(134.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0165_HFILL	00000056	(86.)	87	(135.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0166_D	00000002	(2.)	88	(136.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0166_T	00000100	(256.)	89	(137.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0166_GDCS	00000043	(67.)	8A	(138.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0166_ERTEMP	00000009	(9.)	8B	(139.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0166_FCACHE	00000001	(1.)	8C	(140.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0166_HFILL	00000031	(49.)	8D	(141.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0167_D	00000002	(2.)	8E	(142.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0167_T	00000180	(384.)	8F	(143.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0167_ERTEMP	00000011	(17.)	90	(144.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0167_FCACHE	00000009	(9.)	91	(145.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0167_GDCS	00000001	(1.)	92	(146.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0167_HFILL	00000063	(99.)	93	(147.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0168_D	00000002	(2.)	94	(148.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0168_T	00000100	(256.)	95	(149.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0168_GDCS	0000002D	(45.)	96	(150.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0168_ERTEMP	00000001	(1.)	97	(151.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0168_FCACHE	00000001	(1.)	98	(152.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0168_HFILL	0000004F	(79.)	99	(153.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0169_D	00000002	(2.)	9A	(154.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0169_T	00000080	(128.)	9B	(155.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0169_GDCS	0000006F	(111.)	9C	(156.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0169_ERTEMP	0000000D	(13.)	9D	(157.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE

X_0169_FCACHE	0000000D	(13.)	9E (158.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0169_HFILL	00000075	(117.)	9F (159.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0170_D	00000002	(2.)	A0 (160.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0170_T	00000180	(384.)	A1 (161.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0170_ERTEMP	00000025	(37.)	A2 (162.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0170_FCACHE	00000001	(1.)	A3 (163.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0170_GDCS	00000001	(1.)	A4 (164.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0170_HFILL	00000057	(87.)	A5 (165.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0171_D	00000002	(2.)	A6 (166.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0171_T	00000100	(256.)	A7 (167.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0171_GDCS	0000004E	(78.)	A8 (168.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0171_ERTEMP	00000025	(37.)	A9 (169.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0171_FCACHE	00000001	(1.)	AA (170.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0171_HFILL	0000000A	(10.)	AB (171.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0172_D	00000002	(2.)	AC (172.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0172_T	00000100	(256.)	AD (173.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0172_GDCS	0000002D	(45.)	AE (174.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0172_ERTEMP	00000001	(1.)	AF (175.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0172_FCACHE	00000001	(1.)	B0 (176.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0172_HFILL	0000004F	(79.)	B1 (177.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0173_D	00000002	(2.)	B2 (178.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0173_T	00000100	(256.)	B3 (179.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0173_GDCS	00000022	(34.)	B4 (180.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0173_ERTEMP	00000001	(1.)	B5 (181.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0173_FCACHE	00000001	(1.)	B6 (182.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0173_HFILL	0000005A	(90.)	B7 (183.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0174_D	00000002	(2.)	B8 (184.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0174_T	00000100	(256.)	B9 (185.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0174_GDCS	00000022	(34.)	BA (186.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0174_ERTEMP	00000001	(1.)	BB (187.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0174_FCACHE	00000001	(1.)	BC (188.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0174_HFILL	0000005A	(90.)	BD (189.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0175_D	00000002	(2.)	BE (190.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0175_T	00000180	(384.)	BF (191.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0175_GDCS	00000038	(56.)	C0 (192.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0175_ERTEMP	0000000D	(13.)	C1 (193.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0175_FCACHE	00000001	(1.)	C2 (194.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0175_HFILL	00000038	(56.)	C3 (195.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0176_D	00000002	(2.)	C4 (196.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0176_T	00000080	(128.)	C5 (197.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0176_GDCS	00000022	(34.)	C6 (198.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0176_ERTEMP	00000001	(1.)	C7 (199.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0176_FCACHE	00000001	(1.)	C8 (200.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0176_HFILL	0000005A	(90.)	C9 (201.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0177_D	00000002	(2.)	CA (202.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0177_T	00000080	(128.)	CB (203.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0177_GDCS	00000043	(67.)	CC (204.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0177_ERTEMP	00000001	(1.)	CD (205.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0177_FCACHE	00000001	(1.)	CE (206.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0177_HFILL	00000039	(57.)	CF (207.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0178_D	00000002	(2.)	D0 (208.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0178_T	00000080	(128.)	D1 (209.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0178_GDCS	00000038	(56.)	D2 (210.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0178_ERTEMP	00000001	(1.)	D3 (211.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0178_FCACHE	00000001	(1.)	D4 (212.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE

```
X_0178_HFILL            00000044  (   68.)  D5 (213.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0179_D                00000002  (    2.)  D6 (214.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0179_T                00000080  (  128.)  D7 (215.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0179_GDCS             0000002D  (   45.)  D8 (216.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0179_ERTEMP           00000001  (    1.)  D9 (217.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0179_FCACHE           00000001  (    1.)  DA (218.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0179_HFILL            0000004F  (   79.)  DB (219.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0180_D                00000002  (    2.)  DC (220.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0180_T                00000100  (  256.)  DD (221.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0180_GDCS             00000043  (   67.)  DE (222.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0180_ERTEMP           00000001  (    1.)  DF (223.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0180_FCACHE           00000001  (    1.)  E0 (224.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0180_HFILL            00000039  (   57.)  E1 (225.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0181_D                00000002  (    2.)  E2 (226.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0181_T                00000080  (  128.)  E3 (227.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0181_GDCS             0000002D  (   45.)  E4 (228.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0181_ERTEMP           00000001  (    1.)  E5 (229.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0181_FCACHE           00000001  (    1.)  E6 (230.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0181_HFILL            0000004F  (   79.)  E7 (231.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0182_D                00000000  (    0.)  E8 (232.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
```

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	28	00:00:00.17	00:00:00.56
Command processing	845	00:00:00.90	00:00:02.34
Pass 1	4606	00:02:18.96	00:05:19.30
Symbol table sort	1	00:00:00.76	00:00:01.81
Pass 2	4477	00:00:40.65	00:01:27.67
Symbol table output	2	00:00:00.21	00:00:00.36
Psect synopsis output	51	00:00:00.83	00:00:01.68
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	10011	00:03:02.49	00:06:53.83

The working set limit was 4096 pages.
1792105 bytes (3501 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 367 non-local and 295 local symbols.
8723 source lines were read in Pass 1, producing 497 object records in Pass 2.
99 pages of virtual memory were used to define 46 macros.

Macro library statistics

Macro library name	Macros defined
DISK$UCODPACK00:[VAX750]COMETMAC.MLB;805	46
SYS$COMMON:[SYSLIB]STARLET.MLB;2	0
TOTALS (all libraries)	46

1899 GETS were required to define 46 macros.

```
ZZ-ECKAB-9.2    VAX-11 Macro Run Statistics                    13-MAY-1987       Fiche 5 Frame I7          Sequence 910
ECKAB                          VAX-11/750 DPM MICRO DIAGNOSTICS    11-DEC-1985 09:45:22  VAX/VMS Macro V04-00        Page 18
VAX-11 Macro Run Statistics                                        11-DEC-1985 09:44:57  [VAX750.DPM]ECKAB4.MAR;1          (1

There were 0 errors, 1 warning and 0 information messages, on lines:
 4736 (1)

MACRO UCOD$0:[VAX750.DPM]TITLE+DISK$UCODPACK00:[VAX750]COMETASM+UCOD$0:[VAX750.DPM]ECKAB4/LIS=ECKAB4/OBJ=ECKAB4
```

(1) 24 "DPM Revision History"
(1) 1 ""
(1) 3 "DIAGNOSTIC MICROCODE MACROS"
(1) 98 " RDM Control File Macros"
(1) 109 " Diagnostic Macros"
(1) 194 ""
(1) 195 "MODULE GATE ARRAY MAPS"
(1) 197 " L0001 MULTIPLIER AND GATE ARRAY LOCATION"
(1) 254 " L0002 GATE ARRAY LOCATION"
(1) 311 " L0003 GATE ARRAY LOCATION"
(1) 369 " L0004 GATE ARRAY LOCATION"
(1) 427 " L0011 & L0016 GATE ARRAY LOCATION"
(1) 485 ""
(1) 490 "EQUATED SYMBOLS
(1) 493 ""
(1) 2 "TEST 0B6 TEST BRANCHING ON WBUS DATA
(1) 176 "TEST 0B7 TEST STATUS FLAGS WITH WCTRL FUNCTIONS
(1) 282 "TEST 0B8 TEST STEP COUNTER USING WCTRL FUNCTIONS
(1) 388 "TEST 0B9 BRANCH ON STATUS FLAGS TEST 1
(1) 590 "TEST 0BA BRANCH ON STATUS FLAGS TEST 2
(1) 746 "TEST 0BB TEST MISC STEP COUNTER FUNCTIONS
(1) 863 "TEST 0BC BRANCH ON STEP COUNTER TEST 1
(1) 966 "TEST 0BD BRANCH ON STEP COUNTER TEST 2
(1) 1111 "TEST 0BE TEST DECREMENTING THE STEP COUNTER
(1) 1216 "TEST 0BF TEST BRANCH ON FPD
(1) 1317 "TEST 0C0 TEST BRANCH ON MMNOINT
(1) 1444 "TEST 0C1 CLEAR STEP COUNTER AND FLAGS ON IRD
(1) 1582 "TEST 0C2 IRD1 Instruction Address Decode Test 1
(1) 2065 "TEST 0C3 IRD1 Instruction Address Decode Test 2
(1) 3497 "TEST 0C4 IRD1 Instruction Address Decode Test 3
(1) 3991 "TEST 0C5 IRD1 Instruction Address Decode Test 4
(1) 5435 "TEST 0C6 First IRDX Instruction Address Decode Test 1
(1) 6091 "TEST 0C7 First IRDX Instruction Address Decode Test 2
(1) 7700 "TEST 0C8 First IRDX Instruction Address Decode Test 3
(1) 8356 "TEST 0C9 First IRDX Instruction Address Decode Test 4
(1) 9965 "TEST 0CA Second IRDX Instruction Address Decode Test 1
(1) 10661 "TEST 0CB Second IRDX Instruction Address Decode Test 2
(1) 12311 "TEST 0CC Second IRDX Instruction Address Decode Test 3
(1) 13006 "TEST 0CD Second IRDX Instruction Address Decode Test 4

```
0000     1          .TITLE  ECKAB VAX-11/750 DPM MICRO DIAGNOSTICS
0000     2          .IDENT  /V09.01/
0000     3 ;
0000     4 ; COPYRIGHT (C) 1980,1983
0000     5 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     6 ;
0000     7 ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE  ONLY ON A SINGLE
0000     8 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000     9 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    10 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    11 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    12 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    13 ; REMAIN IN DEC.
0000    14 ;
0000    15 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    16 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    17 ; CORPORATION.
0000    18 ;
0000    19 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    20 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    21 ;
0000    22 ;
```

```
0000     24          .SBTTL  "DPM Revision History"
0000     25 ;++
0000     26 ; DPM MICRO-DIAGNOSTIC PACKAGE
0000     27 ;
0000     28 ; REVISION HISTORY
0000     29 ;
0000     30 ; 00-01 Tom Groetzinger 4-FEB-1980
0000     31 ;       Started file.
0000     32 ;
0000     33 ; 00-02 Dave Spain      27-FEB-1980
0000     34 ;       Added CS BUS INTEGRITY test.
0000     35 ;
0000     36 ; 00-03 Tom Groetzinger 12-MAR-1980
0000     37 ;       Added five tests.
0000     38 ;
0000     39 ; 00-04 Tom Groetzinger 17-MAR-1980
0000     40 ;       Added SRK,SRM callouts to some tests.
0000     41 ;
0000     42 ; 00-05 Tom Groetzinger 17-MAR-1980
0000     43 ;       Improved tests 7 and 16 to increase coverage due to Westboro feedback
0000     44 ;
0000     45 ; 00-06 Tom Groetzinger 17-MAR-1980
0000     46 ;       Debugged test 16 dueu to major rewrite
0000     47 ;
0000     48 ; 00-07 Tom Groetzinger 17-MAR-1980
0000     49 ;       Added 9 Super Rotator tests
0000     50 ;
0000     51 ; 01-08 Dave Spain       4-APR-80
0000     52 ;       Added 4 Micro-sequencer tests for offical release.
0000     53 ;
0000     54 ; 01-09 Bill Landry     4-APR-1980
0000     55 ;       Added the XB, PREFETCH and IRD tests from ECKAC.SRC so
0000     56 ;       manufacturing can run one tape for DPM.
0000     57 ;
0000     58 ; 00-10 Bill Landry     4-APR-1980
0000     59 ;       Let's save version 1.0 for our first SDC release.
0000     60 ;
0000     61 ; 00-11 Tom Groetzinger 7-APR-1980
0000     62 ;       Added 2 ALK Tests
0000     63 ;
0000     64 ; 00-12 Tom Groetzinger 8-APR-1980
0000     65 ;       Deleted last 2 D Register outputs from ALK Test #6
0000     66 ;
0000     67 ; 00-13 Tom Groetzinger 14-APR-1980
0000     68 ;       Added D Register output to ALK test #6
0000     69 ;
0000     70 ; 00-14 Dave Spain      25-APR-1980
0000     71 ;       Changed mask on the micro-sequencer tests to ignore bit 13 of the
0000     72 ;       Control Store address.
0000     73 ;
0000     74 ; 00-15 Dave Spain      28-APR-1980
0000     75 ;       Fixed bug in ALK Tests 6 and 7 by replacing VSIZE/1 micro order with a
0000     76 ;       SIZE[LONG] macro in micro words 6 and 3 respectively.
0000     77 ;
0000     78 ; 00-16 Dave Spain      29-APR-1980
```

```
0000     79 ;       Changed mask on CS address for Instruction Address Decode test from
0000     80 ;       3FFF to 1FFF to ignore bit 13 of Control Store address.
0000     81 ;
0000     82 ; 00-17 Dave Spain      30-APR-1980
0000     83 ;       Added CCS module to error callout for Control Store Bus Integrity test.
0000     84 ;
0000     85 ; 00-18 Dave Spain       9-MAY-1980
0000     86 ;       Added SRM gate array to error callout routine in RTMP0 and RTEMP
0000     87 ;       Sratchpad tests.
0000     88 ;
0000     89 ; 00-19 Bill Landry     10-JUN-1980
0000     90 ;       Added RDM tests to beginning of tape
0000     91 ;
0000     92 ; 01-00 Bill Landry/Dave Spain  13-JUN-1980
0000     93 ;       Added macro file to title section and compatibility mode IRD tests
0000     94 ;       then submitted to RELEASE ENGINERRING.
0000     95 ;
0000     96 ; 01-01 Dave Spain      19-JUN-1980
0000     97 ;       Fixed prefetch incrementer test to clear out cache data after every
0000     98 ;       prefetch. This will prevent the test from mistaking bad addresses from
0000     99 ;       good ones.
0000    100 ;
0000    101 ; 01-02 Dave Spain       2-JUL-1980
0000    102 ;       Added CM Second IRDX tests to file.
0000    103 ;
0000    104 ; 01-03 Dave Spain      18-AUG-1980
0000    105 ;       Added CM branch test to file.
0000    106 ;
0000    107 ; 01-04 Dave Spain      10-SEPT-1980
0000    108 ;       Fixed "DEFAULT MTEMP0 TEST"
0000    109 ;
0000    110 ; 01-05 Bill Landry     11-SEPT-1980
0000    111 ;       Revised RDM to CMI tests
0000    112 ;
0000    113 ; 02-00 Dave Spain      28-SEPT-1980
0000    114 ;       Added rotator tests to file and submitted to SDC.
0000    115 ;
0000    116 ; 02-01 Dave Spain      28-OCT-1980
0000    117 ;       Added QV flag to tests to be run by manufacturing.
0000    118 ;
0000    119 ; 02-02 Bill Landry     10-NOV-1980
0000    120 ;       Added tests for TOK and SAC gate arrays.
0000    121 ;
0000    122 ; 02-03 Bill Landry     13-NOV-1980
0000    123 ;       Added test of REI CHECK
0000    124 ;
0000    125 ; 03-00 Dave Spain      17-NOV-1980
0000    126 ;       Added rotator tests 23-28 and released to SDC.
0000    127 ;
0000    128 ; 03-01 Dave Spain      10-DEC-1980
0000    129 ;       Added rotator tests 29-42.
0000    130 ;
0000    131 ; 03-02 Bill Landry     11-DEC-1980
0000    132 ;       Fixed bug in CS ADDRESS test
0000    133 ;
```

```
0000   134 ; 03-03 Dave Spain      12-DEC-1980
0000   135 ;       Enhanced Rotator Test 1 to call out ALP slices.
0000   136 ;
0000   137 ; 03-04 Bill Landry     16-DEC-1980
0000   138 ;       Added DSIZE ROM tests
0000   139 ;
0000   140 ; 03-05 Bill Landry     31-DEC-1980
0000   141 ;       Last great act of 80. Separated DPM and RDM tests into two files.
0000   142 ;
0000   143 ; 04-00 Bill Landry     5-JAN-1981
0000   144 ;       Submitted to RELEASE ENGINERRING
0000   145 ;
0000   146 ; 04-01 Bill Landry     14-JAN-1981
0000   147 ;       Revised INTERVAL TIMER INTERRUPT TEST to fix bug caused by INT PEND
0000   148 ;       ECO
0000   149 ;
0000   150 ; 04-02 Dave Spain      16-JAN-1981
0000   151 ;       Fixed error callout in CS Parity Error, XB, and Prefetch Incrementer
0000   152 ;       tests and added seven new rotator tests.
0000   153 ;
0000   154 ; 04-03 Bill Landry     4-FEB-1981
0000   155 ;       Added tests for CCC gate array
0000   156 ;
0000   157 ; 04-04 Dave Spain      9-FEB-1981
0000   158 ;       Changed the names of rotator tests 23-33.
0000   159 ;
0000   160 ; 04-05 Bill Landry     24-FEB-1981
0000   161 ;       Added tests for DPM discrete branching logic.
0000   162 ;
0000   163 ; 04-06 Bill Landry     27-FEB-1981
0000   164 ;       Added tests for branching logic in PHB gate array.
0000   165 ;
0000   166 ; 05-00 Dave Spain       1-MAR-1981
0000   167 ;       Added 5 SRK status rotator tests and submitted to SDC.
0000   168 ;
0000   169 ; 05-01 Dave Spain      25-MAR-1981
0000   170 ;       Fixed a loop on test bug in the Alu Functions Test by adding a LDFIELD
0000   171 ;       pseudo-op that reinitializes the ROT field in Subtest 1.
0000   172 ;
0000   173 ; 05-02 Bill Landry     25-MAR-1981
0000   174 ;       Fixed bug in BUT/NO.FPA test.  I was branching on the wrong conditions.
0000   175 ;
0000   176 ; 05-03 Dave Spain      14-APR-1981
0000   177 ;       Cleaned up documentation for R Scratchpads Write All Cells Test.
0000   178 ;
0000   179 ; 06-00 Bill Landry     2-MAY-1981
0000   180 ;       Added SPA branching, CON registers, TU58 registers, and CCS PARITY
0000   181 ;       tests.  Submitted to Release Engineering.
0000   182 ;
0000   183 ; 06-01 Dave Spain      17-JUN-1981
0000   184 ;       Replaced the old IRD1 Instruction Address Decode test with two new
0000   185 ;       tests that toggle the PSL's FPD bit and can handle an FPA.
0000   186 ;
0000   187 ; 06-02 Bill Landry     19-AUG-1981
0000   188 ;       Fixed bugs created by introduction of FPA option.
```

```
0000   189 ;       Replaced all references to COMET with the word CPU.
0000   190 ;       Fixed typing error in vbus data of CPU CLOCK TEST.
0000   191 ;       Submitted to RELEASE ENGINEERING.
0000   192 ;
0000   193 ; 06-03 Bill Landry     24-SEPT-1981
0000   194 ;       Fixed potential clock problem with FPA in Z BIT TEST 1.
0000   195 ;
0000   196 ; 06-04 Bill Landry     9-OCT-1981
0000   197 ;       Added changes for ECO L0002-TW002
0000   198 ;
0000   199 ; 07-00 Bill Landry     31-OCT-1981
0000   200 ;       Removed changes for ECO L0002-TW002.  ECO is on hold.
0000   201 ;       Added tests for FPA to CPU interface logic
0000   202 ;       Moved WCS test to MIC tape
0000   203 ;       Submitted to RELEASE ENGINEERING.
0000   204 ;
0000   205 ; 07-01 Bill Landry     13-NOV-1981
0000   206 ;       Stopped release of 7.0
0000   207 ;       Added FPA TRAP LOGIC TEST
0000   208 ;       Enhanced error messages for FPA
0000   209 ;       Resubmitted to RELEASE ENGINEERING
0000   210 ;
0000   211 ; 07-02 Bill Landry     20-NOV-1981
0000   212 ;       Stopped release of 7.1
0000   213 ;       Enhanced FPA ENABLE/DISABLE TEST to check for MSEQ INIT
0000   214 ;       Resubmitted to RELEASE ENGINEERING
0000   215 ;
0000   216 ; 07-03 Dave Shull      26-Mar-1982
0000   217 ;       Added the ERRLOOP pseudo to the Rotator Extract tests which were
0000   218 ;       missing these and couldn't be looped on.
0000   219 ;
0000   220 ; 07-04 Dave Shull      05-Apr-1982
0000   221 ;       Removed all BEGINSA and ENDSA pseudo's.
0000   222 ;       Submitted to RELEASE ENGINEERING
0000   223 ;
0000   224 ; 07-05 Dave Shull      21-May-1982
0000   225 ;       Add a subtest to the Micro Sequencer Stack Test that will test bits
0000   226 ;       11 and 12 of the micro stack.
0000   227 ;
0000   228 ; 07-06 Dave Shull      24-May-1982
0000   229 ;       Changed documentation in the CM Odd Address BUT Test to reflect the
0000   230 ;       ECO L0002-TW002
0000   231 ;
0000   232 ; 08-00 Dave Shull      18-Jun-1982
0000   233 ;       Added 10 new tests.  Two of those tests are an addition to the existing
0000   234 ;       IRD1 rom tests.  These 2 new tests check the signal ROM OS INH H.
0000   235 ;       The other 8 new tests check the validity of the IRDX ROMs.  They check
0000   236 ;       both the address and the signal ROM OS INH H.
0000   237 ;
0000   238 ; 08-01 Dave Shull      22-Jun-1982
0000   239 ;       Added a subtest to the Register Backup Stack test that will check the
0000   240 ;       bit that records Increment/Decrement of the GPR.
0000   241 ;
0000   242 ; 09-00 Dave Shull      03-Nov-1982
0000   243 ;       Added two new Tests for coverage the new PCS (L0008) module.
```

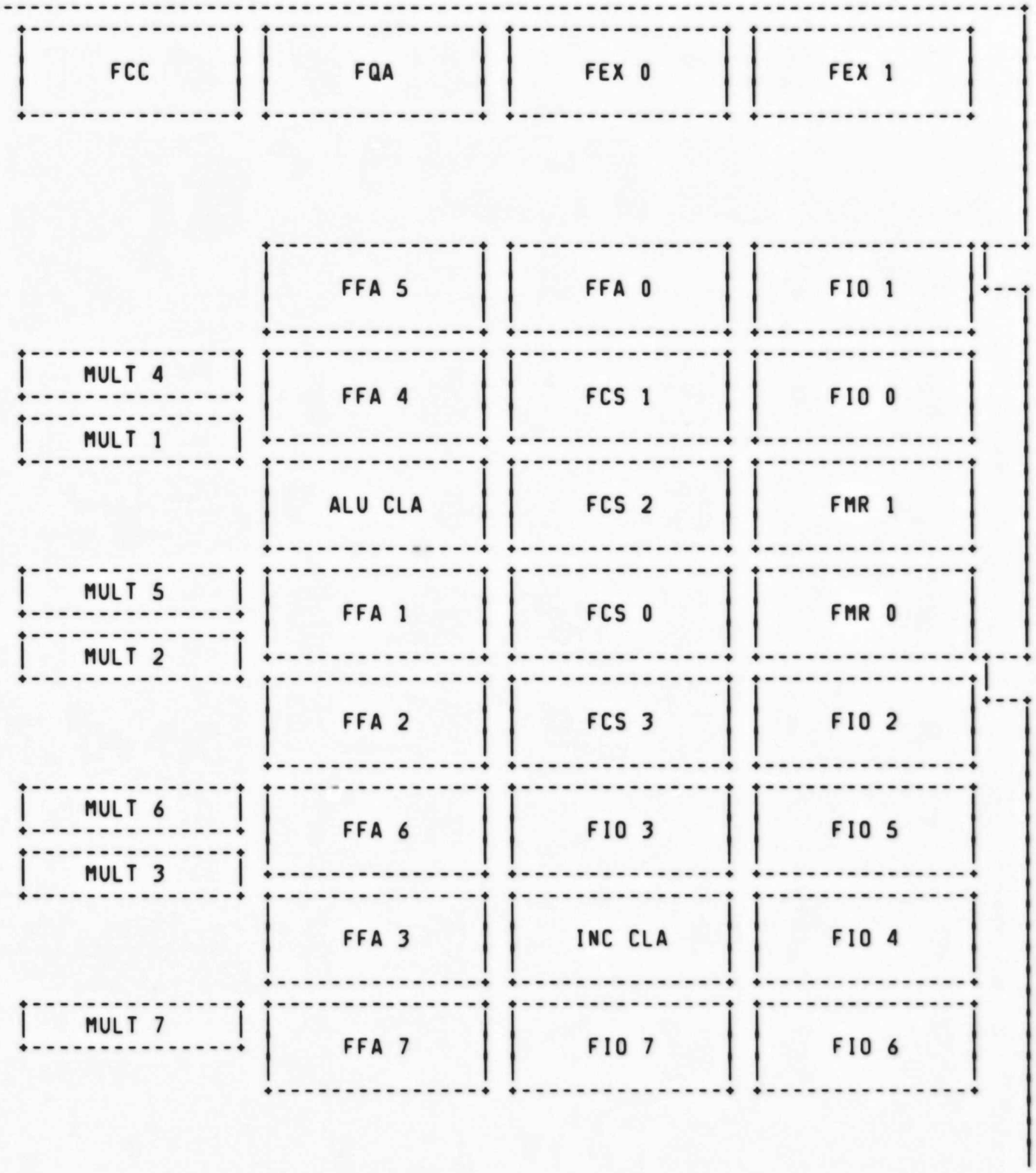
```
                        0000    244 ;
                        0000    245 ; 09-01 Dave Smith      15-May-1983
                        0000    246 ;       Correct DCS in ALK test 3,6,7. Disabled ALP's on the WBUS.
                        0000    247 ;--
                        0000      1         .SBTTL  ""
```

```
0000     3          .SBTTL  "DIAGNOSTIC MICROCODE MACROS"
0000     4 ;++
0000     5 ; Revision History
0000     6 ;
0000     7 ; 28    Dave Spain      22-Jun-82
0000     8 ;       Removed LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
0000     9 ;       Decided not to use them.
0000    10 ;
0000    11 ; 27    Dave Spain      15-Jun-82
0000    12 ;       Added LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
0000    13 ;
0000    14 ; 26    Dave Spain      19-May-82
0000    15 ;       Added M[]_FPA macro.
0000    16 ;
0000    17 ; 25    Rich Lewis      23-Feb-82
0000    18 ;       Added PSL_WB, M[]_R[].OR.NIB0(M), WB_FPA CCR, WB_FPA TCR, Q_R[].ASL.P,
0000    19 ;       RDM_M[].XZ.OR.R[], FPA_WB macros
0000    20 ;
0000    21 ; 24    Dave Spain      02-Feb-82
0000    22 ;       Added FPA_R[]+M[] macro.
0000    23 ;
0000    24 ; 23    Bill Landry     22-OCT-81
0000    25 ;       Added PSL_RDM macro
0000    26 ;
0000    27 ; 22    Dave Spain       5 Oct 81
0000    28 ;       Added RDM_FPA and FPA_M[] FPA_RDM macros.
0000    29 ;
0000    30 ; 21    Bill Landry     29 Sept 81
0000    31 ;       Added FPA macro, FPA_M[] FPA_R[].
0000    32 ;
0000    33 ; 20    Dave Spain      14 Aug 81
0000    34 ;       Added the TESTFPA [] macro.
0000    35 ;
0000    36 ; 19    Dave Spain      18 July 81
0000    37 ;       Added the following FPA macros, FPA_RDM and FPA_R[].
0000    38 ;
0000    39 ; 18    Dave Spain      24 Sept 80
0000    40 ;       Added RDM_Q macro. This macro uses the ALU.
0000    41 ;
0000    42 ; 17    Bill Landry     23 Sept 80
0000    43 ;       Added macro VA_VA-ZLIT0[] for testing memory in decending order.
0000    44 ;
0000    45 ; 16    Dave Spain      27 August 80
0000    46 ;       Changed RDM_Q macro to RDM_Q Q_D. This more accurately reflects macro.
0000    47 ;
0000    48 ; 15    Bill Landry     4 June 80
0000    49 ;       Deleted BUS/PRB.RD.PTE from macro CLRTB.VA_VA and CLRCH.VA_VA to
0000    50 ;       satisfy validity checks.
0000    51 ;
0000    52 ; 14    Dave Spain      23 May 80
0000    53 ;       Added BUS/PRB.RD.PTE to the clear cache and clear TB macros for Mr.
0000    54 ;       Bill.
0000    55 ;
0000    56 ; 13    Dave Spain      14 Apr 80
0000    57 ;       Added SIZE[] or VSIZE/1 to some macros. This allows version 65 of
```

```
0000     58 ;       the COMET micro-code defines to run.
0000     59 ;
0000     60 ; 12    Bill Landry      1 Apr 80
0000     61 ;       Added BUS/PRB.RD micro order to RDM_TB macro to compensate for timing
0000     62 ;       problem in ADD gate array.  This is not an APRIL FOOL!
0000     63 ;
0000     64 ; 11    Dave Spain      27 Mar 80
0000     65 ;       Fixed TB macro's to include MSRC/VA to prevent validity failure.
0000     66 ;
0000     67 ; 10    Bill Landry     19 Mar 80
0000     68 ;       Added Q_R[].RL.P and VA_Q.OR.R[]
0000     69 ;
0000     70 ; 09    Bill Landry     19 Mar 80
0000     71 ;       Deleted TB_R[].RL.P.OR.D
0000     72 ;
0000     73 ; 08    Bill Landry     18 Mar 80
0000     74 ;       Added VA_R[].RL.P
0000     75 ;
0000     76 ; 07    Bill Landry     07 Mar 80
0000     77 ;       Added Q_M[].RL.PTE
0000     78 ;
0000     79 ; 06    Dave Spain      06 Mar 80
0000     80 ;       Changed TB_R[].RL.P macro to TB_R[].RL.P.OR.D.
0000     81 ;
0000     82 ; 05    Dave Spain      03 Mar 80
0000     83 ;       Added TB_R[].RL.P macro for Bill Landry.
0000     84 ;
0000     85 ; 04    Dave Spain      22 Feb 80
0000     86 ;       Added CLRTB.VA_R[] macro for Bill Landry.
0000     87 ;
0000     88 ; 03    Dave Spain      30 Jan 80
0000     89 ;       Added Tom Groetzinger's macro definitions.
0000     90 ;
0000     91 ; 02    Dave Spain      30 Jan 80
0000     92 ;       Added Tony Vezza's macro definitions.
0000     93 ;
0000     94 ; 01    Dave Spain      29 Jan 80
0000     95 ;       Macros initiated.
0000     96 ;--
0000     97
0000     98          .SBTTL  "       RDM Control File Macros"
0000     99
0000    100 ;STROBE.V.BUS            "RDMCF0/V.STROBE"
0000    101 ;DCS.ADDR_0              "RDMCF1/DCS.ADDR.CLR"
0000    102 ;STOP.CLOCK              "RDMCF2/STOP.CPU.CLOCK"
0000    103 ;LATCH.CS.ADDR           "RDMCF3/DIS.STROBE.CS.A"
0000    104 ;RDM_WB                  "RDMCF4/H.REG.FROM.WB"
0000    105 ;WB_RDM                  "RDMCF5/WB.FROM.D.REG"
0000    106 ;RDM_CMI                          "RDMCF6/STROBE.CMI"
0000    107 ;INH.CLOCK.SA            "RDMCF7/INH.SA.CLOCK"
0000    108
0000    109          .SBTTL  "       Diagnostic Macros"
0000    110
0000    111 ;CLOCK.EXT                        "CLKX/XTND"
0000    112 ;CLRCH.VA_RDM                     "WB_RDM,WCTRL/CLRCH.VA_WB,BUS/PRB.RD.PTE"
```

```
0000   113 ;CLRCH.VA_VA                    "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRCH.VA_WB
0000   114 ;CLRTB.VA_D                     "RSRC/ZERO,MUX/D.R1,ALU/OR,WCTRL/CLRTB.VA_WB,
0000   115 ;                                       BUS/PRB.RD.PTE"
0000   116 ;CLRTB.VA_RDM                   "WB_RDM,WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
0000   117 ;CLRTB.VA_R[]                   "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/CLRTB.VA_WB,
0000   118 ;                                       BUS/PRB.RD.PTE"
0000   119 ;CLRTB.VA_VA                    "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRTB.VA_WB
0000   120 ;CLRTB.VA_WB                    "WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
0000   121 ;D_LONLIT                       "RSRC/LONLIT,ALPCTL/WX_D_R.Q_D"
0000   122 ;D_M[]_LONLIT                   "D_LONLIT,MSRC/a1"
0000   123 ;D_VA_0                         "RSRC/ZERO,ROT/ZERO,MUX/R.S,ALU/OR,DQ1/D_WX,
0000   124 ;                                       WCTRL/VA_WB"
0000   125 ;FPA_M[] FPA_RDM                        "MSRC/a1,WB_RDM,FPA/FPA_MBUS.FPA_WBUS"
0000   126 ;FPA_M[] FPA_R[]                       "FPA/FPA_MBUS.FPA_WBUS,MSRC/a1,RSRC/a2,MUX/R
0000   127 ;                                       ROT/ZERO,ALU/OR"
0000   128 ;FPA_RDM                              "WB_RDM,FPA/FPA_DATA.WBUS"
0000   129 ;FPA_R[]                             "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,FPA/FPA_DAT
0000   130 ;FPA_R[]+M[]                        "RSRC/a1,MSRC/a2,MUX/M.R1,ALU/A+B+CI,ALUCI/ZERO,FPA/
0000   131 ;FPA_WB                            "FPA/FPA_DATA.WBUS"
0000   132 ;MEMSCAR_R[]                       "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/MEMSCAR_WB"
0000   133 ;M[]_FPA                           "MSRC/a1,SPW/MLONG,FPA/WBUS_FPA"
0000   134 ;M[]_LONLIT                       "MSRC/a1,SPW/MLONG,ALPCTL/WX_R.Q_D,RSRC/LONLIT"
0000   135 ;M[]_R[].OR.NIB0(M)               "MSRC/a1,RSRC/a2,SPW/MLONG,ROT/GETNIB,ALU/OR,
0000   136 ;                                       MUX/R.S,DQ1/NOP"
0000   137 ;M[]_RDM                           "WB_RDM,SPW/MLONG,MSRC/a1"
0000   138 ;PC_PC+RDM                        "WCTRL/PC_PC+WB,WB_RDM"
0000   139 ;PC_PC+4 MB_XB                    "MSRC/XB,PC_PC+I,ISTRM/ISIZE_DSIZE,SIZE[LONG]"
0000   140 ;PC_PC+4 RDM_XB                   "RSRC/ZERO,MSRC/XB,PC_PC+I,ISTRM/ISIZE_DSIZE,
0000   141 ;                                       MUX/M.R1,ALU/OR,RDM_WB,SIZE[LONG]"
0000   142 ;PC_RDM                           "WB_RDM,WCTRL/PC_WB"
0000   143 ;PSL_RDM                          "CCPSL/PSL_WB.CCBR_ALUS,WB_RDM"
0000   144 ;PSL_WB                           "CCPSL/PSL_WB.CCBR_ALUS"
0000   145 ;Q_D WX_R                         "ALPCTL/WX_R.Q_D"
0000   146 ;Q_LONLIT                         "RSRC/LONLIT,ALPCTL/WX_S.Q_R"
0000   147 ;Q_M[].RL.PTE                     "ALPCTL/WX_Q_S,MSRC/a1,ROT/RL.MM.PTE"
0000   148 ;Q_R[].AND.Q                      "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/AND"
0000   149 ;Q_R[].ASL.P                      "ALPCTL/WX_Q_S,RSRC/a1,ROT/ASL.R.P"
0000   150 ;Q_R[].OR.Q                       "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/OR"
0000   151 ;Q_R[].RL.P                       "ALPCTL/WX_Q_S,RSRC/a1,ROT/RL.RR.P"
0000   152 ;RDM_ALUF                        "ALPCTL/WB_ALUF,RDM_WB"
0000   153 ;RDM_FPA                          "FPA/WBUS_FPA,RDM_WB"
0000   154 ;RDM_LONLIT                      "WB_LONLIT,RDM_WB"
0000   155 ;RDM_LONLIT.Q_M                   "RSRC/LONLIT,ALPCTL/WX_R.Q_M,RDM_WB"
0000   156 ;RDM_M[]                          "MSRC/a1,ROT/ZERO,MUX/M.S,ALU/OR,RDM_WB"
0000   157 ;RDM_M[].OR.R[]                   "WB_M[a1].OR.R[a2],RDM_WB"
0000   158 ;RDM_M[]-R[]                      "WB_M[a1]-R[a2],RDM_WB"
0000   159 ;RDM_M[].XZ.OR.R[]               "ALU/A+B+CI,MUX/R.S,ROT/XZ.MM,MSRC/a1,RSRC/a2,
0000   160 ;                                RDM_WB"
0000   161 ;RDM_PC                          "MSRC/PC,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
0000   162 ;RDM_PCBACK                      "RSRC/ZERO,MSRC/PCBACK,MUX/M.R1,ALU/OR,RDM_WB"
0000   163 ;RDM_Q                           "RSRC/ZERO,MUX/R.Q,ALU/OR,RDM_WB"
0000   164 ;RDM_Q Q_D                       "ALPCTL/WX_Q.Q_D,RDM_WB"
0000   165 ;RDM_R[]                         "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,RDM_WB"
0000   166 ;RDM_TB                          "WB_M[TB],RDM_WB,BUS/PRB.RD,SIZE[LONG]"
0000   167 ;RDM_VA                          "MSRC/VA,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
```

```
0000   168 ;RNUM_LONLIT                    "ALPCTL/WX_D_R.Q_M,MSRC/RNUM_WBUS,RSRC/LONLIT"
0000   169 ;R[]_RDM                              "WB_RDM,SPW/RLONG,RSRC/a1"
0000   170 ;R[].DT_Q                       "RSRC/a1,D_Q Q_D,SPW/RSIZE,VSIZE/1"
0000   171 ;R[]_VA                        "RSRC/a1,MSRC/VA,ROT/ZERO,MUX/M.S,ALU/OR,SPW/RLONG"
0000   172 ;R[]_WB                        "RSRC/a1,SPW/RLONG"
0000   173 ;TB_RDM                        "WCTRL/TB_WB,WB_RDM,MSRC/VA"
0000   174 ;TB_WB                         "WCTRL/TB_WB,MSRC/VA"
0000   175 ;TESTFPA []                    "TESTFPA/a1"
0000   176 ;VA_PC+LONLIT+I.PC_PC+I        "RSRC/LONLIT,MSRC/XB.PC_PC+I,ROT/ZERO,MUX/R.S,ALU/OR
0000   177 ;                                      ISTRM/ISIZE_DSIZE,WCTRL/VA_PC+I+W.PC_PC+I,
0000   178 ;                                      SIZE[LONG]"
0000   179 ;VA_Q.OR.R[]                    "WCTRL/VA_WB,RSRC/a1,ALU/OR,MUX/R.Q"
0000   180 ;VA_RDM                        "WB_RDM,WCTRL/VA_WB"
0000   181 ;VA_R[].RL.P                   "ALPCTL/WX_S,ROT/RL.RR.P,RSRC/a1,WCTRL/VA_WB"
0000   182 ;VA_VA-ZLIT0[]                 "WCTRL/VA_WB,LIT/LITRL,LITRL/a1,ROT/ZLIT0,MSRC/VA,
0000   183 ;                                      MUX/M.S,ALU/A-B-CI"
0000   184 ;VA_WB                         "WCTRL/VA_WB"
0000   185 ;WB_FPA CCR                    "FPA/WBUS_FPA.CC"
0000   186 ;WB_FPA TCR                    "WCTRL/FPTCR"
0000   187 ;WB_LONLIT                     "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
0000   188 ;WB_LONLIT.Q_M                 "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
0000   189 ;WB_M[]+R[]                    "ALU/A+B+CI,ALUCI/ZERO,MUX/M.R1,MSRC/a1,RSRC/a2"
0000   190 ;WB_R[].Q_D                    "RSRC/a1,ALPCTL/WX_R.Q_D"
0000   191 ;WDR_R[]                             "WCTRL/WDR_WB,ALU/OR,MUX/R.S,RSRC/a1,ROT/ZER
0000   192 ;WRITE.LONG R[]                 "BUS/WRITE.LNG,WCTRL/WDR_WB.UR,RSRC/a1,ROT/ZERO,
0000   193 ;                                      MUX/R.S,ALU/OR,SIZE[LONG]"
0000   194         .SBTTL  ""
0000   195         .SBTTL  "MODULE GATE ARRAY MAPS"
```



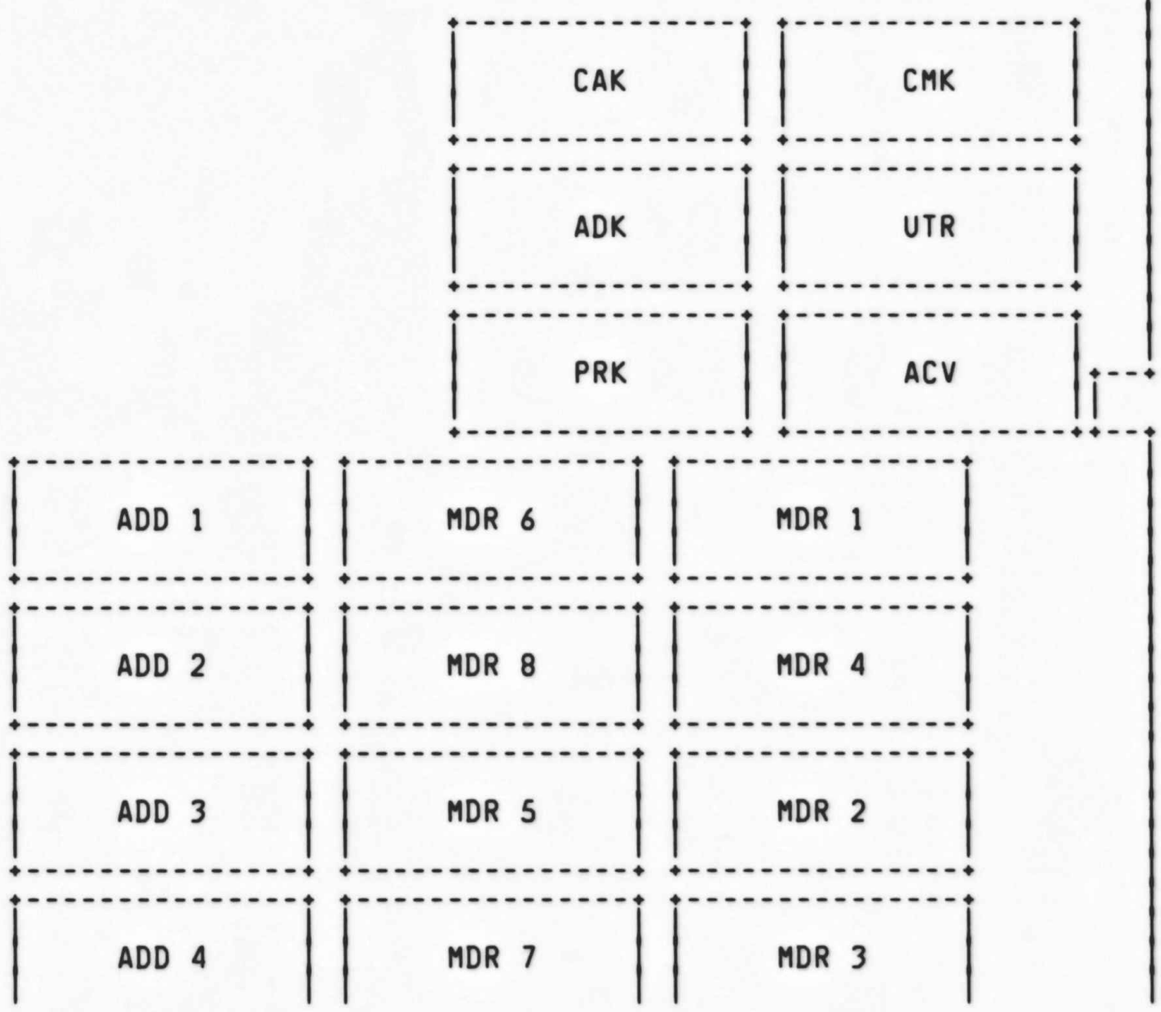
```
0000          .SBTTL  "      L0001 MULTIPLIER AND GATE ARRAY LOCATION"
0000  ;+-----------------------------------------------------------------------------+
0000  ;|  +-------------+  +-------------+  +-------------+  +-------------+   |
0000  ;|  |           |  |           |  |           |  |           |   |
0000  ;|  |    FCC    |  |    FQA    |  |   FEX 0   |  |   FEX 1   |   |
0000  ;|  |           |  |           |  |           |  |           |   |
0000  ;|  +-------------+  +-------------+  +-------------+  +-------------+   |
0000  ;|                                                                       |
0000  ;|                                                                       |
0000  ;|                                                                       |
0000  ;|                                                                       |
0000  ;|                                                                       |
0000  ;|                  +-------------+  +-------------+  +-------------++--+
0000  ;|                  |           |  |           |  |           |  |
0000  ;|                  |   FFA 5   |  |   FFA 0   |  |   FIO 1   |+--+
0000  ;|                  |           |  |           |  |           |   |
0000  ;|                  +-------------+  +-------------+  +-------------+   |
0000  ;|  +-------------+  +-------------+  +-------------+  +-------------+   |
0000  ;|  |  MULT 4   |  |           |  |           |  |           |   |
0000  ;|  +-------------+  |   FFA 4   |  |   FCS 1   |  |   FIO 0   |   |
0000  ;|  +-------------+  |           |  |           |  |           |   |
0000  ;|  |  MULT 1   |  +-------------+  +-------------+  +-------------+   |
0000  ;|  +-------------+  +-------------+  +-------------+  +-------------+   |
0000  ;|                  |           |  |           |  |           |   |
0000  ;|                  |  ALU CLA  |  |   FCS 2   |  |   FMR 1   |   |
0000  ;|                  |           |  |           |  |           |   |
0000  ;|                  +-------------+  +-------------+  +-------------+   |
0000  ;|  +-------------+  +-------------+  +-------------+  +-------------+   |
0000  ;|  |  MULT 5   |  |           |  |           |  |           |   |
0000  ;|  +-------------+  |   FFA 1   |  |   FCS 0   |  |   FMR 0   |   |
0000  ;|  +-------------+  |           |  |           |  |           |   |
0000  ;|  |  MULT 2   |  +-------------+  +-------------+  +-------------++--+
0000  ;|  +-------------+  +-------------+  +-------------+  +-------------+|
0000  ;|                  |           |  |           |  |           |+--+
0000  ;|                  |   FFA 2   |  |   FCS 3   |  |   FIO 2   |   |
0000  ;|                  |           |  |           |  |           |   |
0000  ;|                  +-------------+  +-------------+  +-------------+   |
0000  ;|  +-------------+  +-------------+  +-------------+  +-------------+   |
0000  ;|  |  MULT 6   |  |           |  |           |  |           |   |
0000  ;|  +-------------+  |   FFA 6   |  |   FIO 3   |  |   FIO 5   |   |
0000  ;|  +-------------+  |           |  |           |  |           |   |
0000  ;|  |  MULT 3   |  +-------------+  +-------------+  +-------------+   |
0000  ;|  +-------------+  +-------------+  +-------------+  +-------------+   |
0000  ;|                  |           |  |           |  |           |   |
0000  ;|                  |   FFA 3   |  |  INC CLA  |  |   FIO 4   |   |
0000  ;|                  |           |  |           |  |           |   |
0000  ;|                  +-------------+  +-------------+  +-------------+   |
0000  ;|  +-------------+  +-------------+  +-------------+  +-------------+   |
0000  ;|  |  MULT 7   |  |           |  |           |  |           |   |
0000  ;|  +-------------+  |   FFA 7   |  |   FIO 7   |  |   FIO 6   |   |
0000  ;|                  |           |  |           |  |           |   |
0000  ;|                  +-------------+  +-------------+  +-------------+   |
0000  ;|                                                                       |
0000  ;|                                                                       |
0000  ;|                                                                       |
```

```
                0000   252 ;+----------------------------------------------------------------------------+
```

```
0000          .SBTTL  "      L0002 GATE ARRAY LOCATION"
0000   ;+-----------------------------------------------------------------------+
0000   ;|                                                                       |
0000   ;|                                                                       |
0000   ;|                                                                       |
0000   ;|                                                                       |
0000   ;|                                                                       |
0000   ;|                                                                       |
0000   ;|                                                                       |
0000   ;|                                                                       |
0000   ;|                                                                       |
0000   ;|                  +---------------+ +---------------+                  |
0000   ;|                  |               | |               |                  |
0000   ;|                  |      PHB      | |      MSQ      |                  |
0000   ;|                  |               | |               |                  |
0000   ;|                  +---------------+ +---------------+                  |
0000   ;|                  +---------------+ +---------------+                  +---+
0000   ;|                  |               | |               |                  |   |
0000   ;|                  |      CCC      | |      SAC      |                  +---+
0000   ;|                  |               | |               |                      |
0000   ;|                  +---------------+ +---------------+                      |
0000   ;|  +---------------+ +---------------+ +---------------+                  |
0000   ;|  |               | |               | |               |                  |
0000   ;|  |      SPA      | |      SRK      | |      IRD      |                  |
0000   ;|  |               | |               | |               |                  |
0000   ;|  +---------------+ +---------------+ +---------------+                  |
0000   ;|  +---------------+ +---------------+ +---------------+                  |
0000   ;|  |               | |               | |               |                  |
0000   ;|  |      TOK      | |      CLA      | |      ALK      |                  |
0000   ;|  |               | |               | |               |                  |
0000   ;|  +---------------+ +---------------+ +---------------+                  |
0000   ;|                  +---------------+ +---------------+                  |
0000   ;|                  |               | |               |                  |
0000   ;|                  |      ALP 1    | |      ALP 5    |                  |
0000   ;|                  |               | |               |                  |
0000   ;|                  +---------------+ +---------------+                  +---+
0000   ;|                  +---------------+ +---------------+                  |   |
0000   ;|                  |               | |               |                  +---+
0000   ;|                  |      ALP 2    | |      ALP 6    |                      |
0000   ;|                  |               | |               |                      |
0000   ;|                  +---------------+ +---------------+                      |
0000   ;|                  +---------------+ +---------------+                      |
0000   ;|                  |               | |               |                      |
0000   ;|                  |      ALP 3    | |      ALP 7    |                      |
0000   ;|                  |               | |               |                      |
0000   ;|                  +---------------+ +---------------+                      |
0000   ;|                  +---------------+ +---------------+                      |
0000   ;|                  |               | |               |                      |
0000   ;|                  |      ALP 4    | |      ALP 8    |                      |
0000   ;|                  |               | |               |                      |
0000   ;|                  +---------------+ +---------------+                      |
0000   ;|  +---------------+ +---------------+ +---------------+ +---------------+  |
0000   ;|  |               | |               | |               | |               |  |
0000   ;|  |      SRM 1    | |      SRM 2    | |      SRM 3    | |      SRM 4    |  |
0000   ;|  |               | |               | |               | |               |  |
0000   ;|  +---------------+ +---------------+ +---------------+ +---------------+  |
```

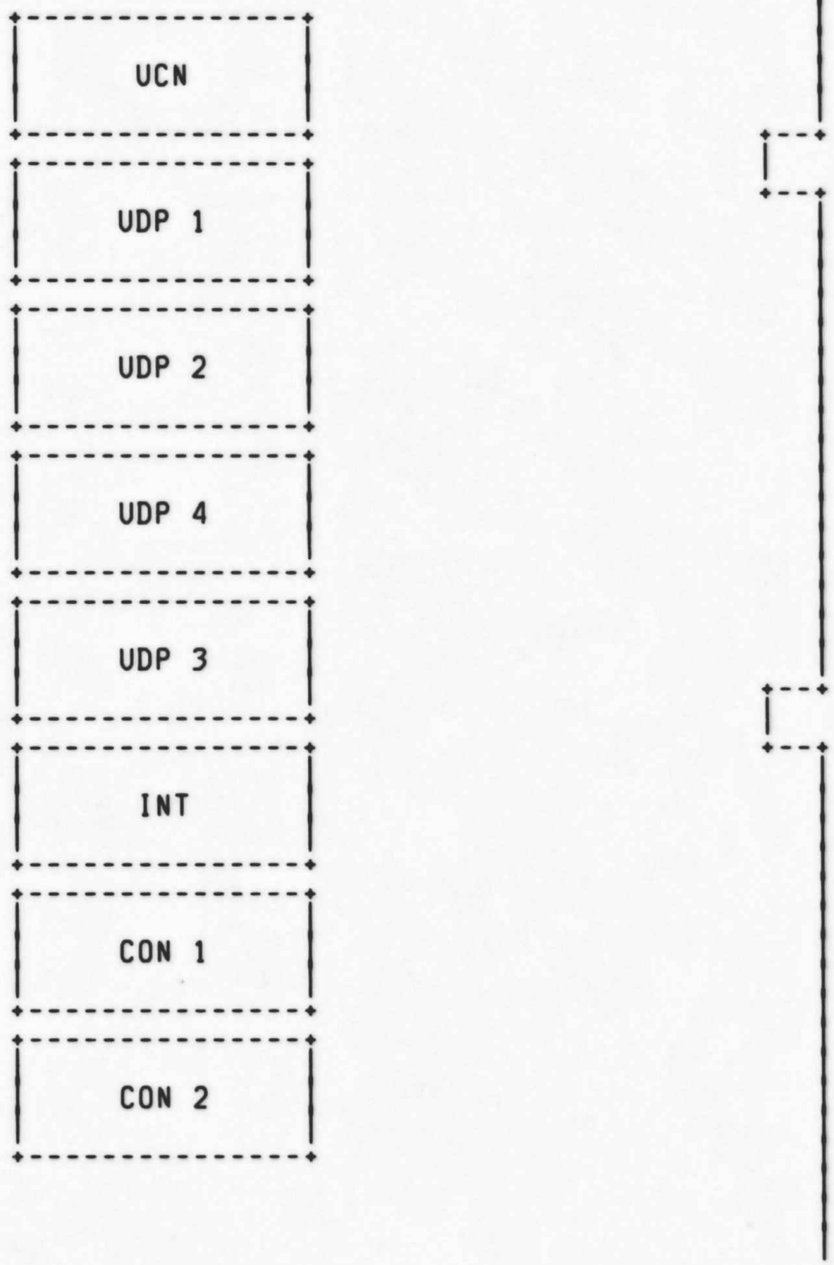
```
                0000   309 ;+-----------------------------------------------------------------------------+
```

```
                        0000   311          .SBTTL  "       L0003 GATE ARRAY LOCATION"
                        0000   312 ;+-----------------------------------------------------------------------+
                        0000   313 ;|                                                                       |
                        0000   314 ;|                                                                       |
                        0000   315 ;|                                                                       |
                        0000   316 ;|                                                                       |
                        0000   317 ;|                                                                       |
                        0000   318 ;|                                                                       |
                        0000   319 ;|                                                                       |
                        0000   320 ;|                                                                       |
                        0000   321 ;|                                                                       |
                        0000   322 ;|                                                                       |
                        0000   323 ;|                                                                       |
                        0000   324 ;|                                                                       |
                        0000   325 ;|                                                                       |
                        0000   326 ;|                                                                       |
                        0000   327 ;|                                                                       |
                        0000   328 ;|                                                                       |
                        0000   329 ;|                                                                       |
                        0000   330 ;|                                                                       |
                        0000   331 ;|                                                                       |
                        0000   332 ;|                                                                       |
                        0000   333 ;|                                                                       |
                        0000   334 ;|                                                                       |
                        0000   335 ;|                                                                       |
                        0000   336 ;|                                                                       |
                        0000   337 ;|                                                                       |
                        0000   338 ;|                                                                       |
                        0000   339 ;|                                                                       |
                        0000   340 ;|                                                                       |
                        0000   341 ;|                                                                       |
                        0000   342 ;|                                                                       |
                        0000   343 ;|                                                                       |
                        0000   344 ;|                                                                       |
                        0000   345 ;|                                                                       |
                        0000   346 ;|                                                                       |
                        0000   347 ;|                                                                       |
                        0000   348 ;|                                                                       |
                        0000   349 ;|                                                                       |
                        0000   350 ;|                                                                       |
                        0000   351 ;|                                                                       |
                        0000   352 ;|                                                                       |
                        0000   353 ;|                                                                       |
                        0000   354 ;|                                                                       |
                        0000   355 ;|                                                                       |
                        0000   356 ;|                                                                       |
                        0000   357 ;|                                                                       |
                        0000   358 ;|                                                                       |
                        0000   359 ;|                                                                       |
                        0000   360 ;|                                                                       |
                        0000   361 ;|                                                                       |
                        0000   362 ;|                                                                       |
                        0000   363 ;|                                                                       |
                        0000   364 ;|                                                                       |
                        0000   365 ;|                                                                       |
```



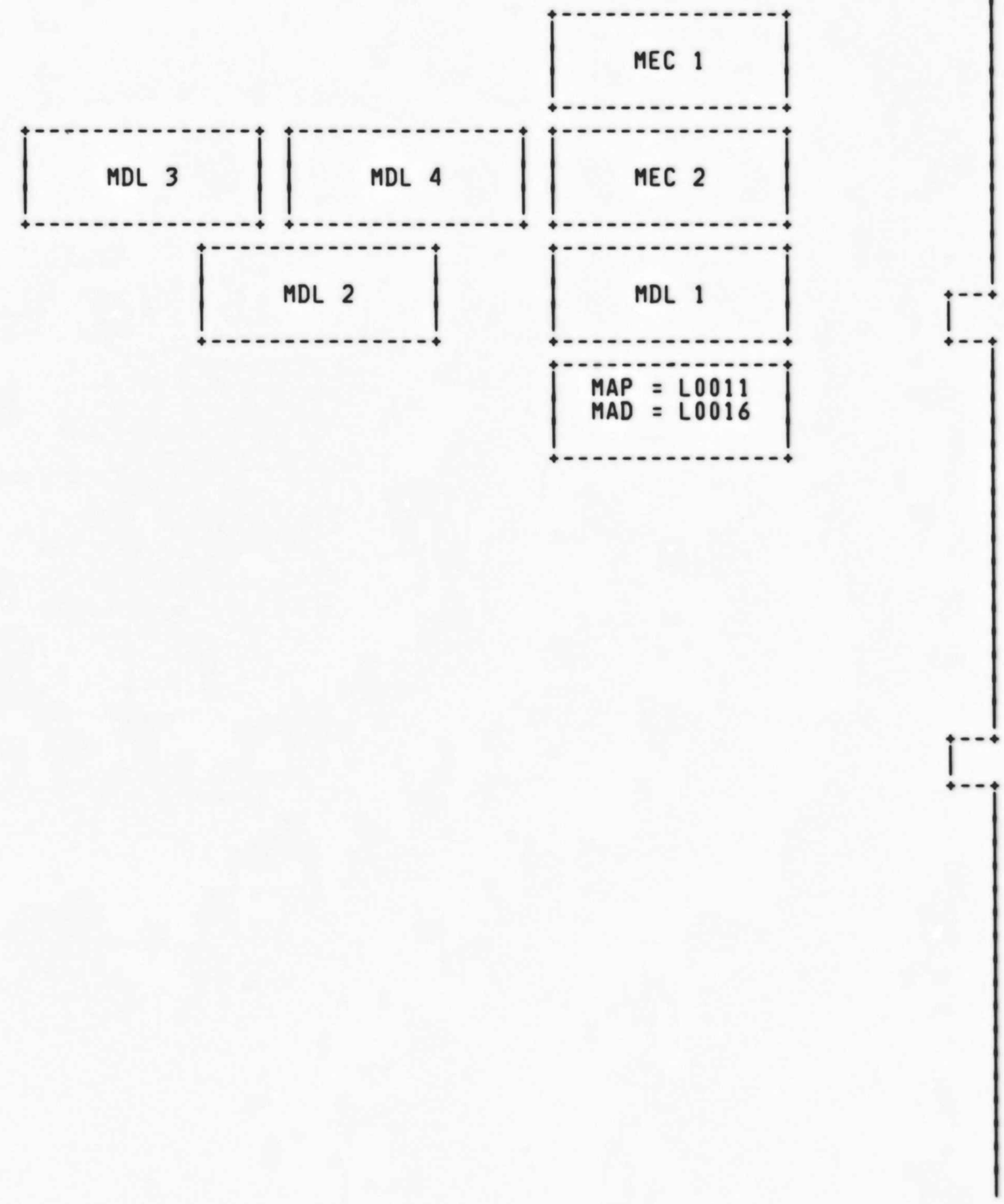
```
0000   366 ;|              +---------------+ +---------------+ +---------------+          |
0000   367 ;+-------------------------------------------------------------------------------+
```

```
                        0000    369          .SBTTL  "       L0004 GATE ARRAY LOCATION"
                        0000    370 ;+-----------------------------------------------------------------------------+
                        0000    371 ;|                                                                             |
                        0000    372 ;|                                                                             |
                        0000    373 ;|                                                                             |
                        0000    374 ;|                                                                             |
                        0000    375 ;|                                                                             |
                        0000    376 ;|                                                                             |
                        0000    377 ;|                                                                             |
                        0000    378 ;|                                                                             |
                        0000    379 ;|                                                                             |
                        0000    380 ;|                                                                             |
                        0000    381 ;|                                                                             |
                        0000    382 ;|                                                                             |
                        0000    383 ;|                                                                             |
                        0000    384 ;|                                                                             |
                        0000    385 ;|                                                                             |
                        0000    386 ;|                                                                             |
                        0000    387 ;|                                                                             |
                        0000    388 ;|                                                                             |
                        0000    389 ;|                                                                             |
                        0000    390 ;|                                                                             |
                        0000    391 ;|                                                                             |
                        0000    392 ;|                                                                             |
                        0000    393 ;|                                                                             |
                        0000    394 ;|                                                                             |
                        0000    395 ;|                                                                             |
                        0000    396 ;|                                                                             |
                        0000    397 ;|                                                                             |
                        0000    398 ;|                                                                             |
                        0000    399 ;|                                                                             |
                        0000    400 ;|                                                                             |
                        0000    401 ;|                                                                             |
                        0000    402 ;|                                                                             |
                        0000    403 ;|                                                                             |
                        0000    404 ;|                                                                             |
                        0000    405 ;|                                                                             |
                        0000    406 ;|                                                                             |
                        0000    407 ;|                                                                             |
                        0000    408 ;|                                                                             |
                        0000    409 ;|                                                                             |
                        0000    410 ;|                                                                             |
                        0000    411 ;|                                                                             |
                        0000    412 ;|                                                                             |
                        0000    413 ;|                                                                             |
                        0000    414 ;|                                                                             |
                        0000    415 ;|                                                                             |
                        0000    416 ;|                                                                             |
                        0000    417 ;|                                                                             |
                        0000    418 ;|                                                                             |
                        0000    419 ;|                                                                             |
                        0000    420 ;|                                                                             |
                        0000    421 ;|                                                                             |
                        0000    422 ;|                                                                             |
                        0000    423 ;|                                                                             |
```



```
ZZ-ECKAB-9.2    " L0004 GATE ARRAY LOCATION"                 13-MAY-1987       Fiche 5 Frame B9          Sequence 929
ECKAB                      VAX-11/750 DPM MICRO DIAGNOSTICS    11-DEC-1985 09:34:58  VAX/VMS Macro V04-00        Page  1
V09.01                     " L0004 GATE ARRAY LOCATION"        14-AUG-1984 15:55:16  [VAX750]COMETASM.MAR;221          (1

                              0000   424 ;|                                                                          |
                              0000   425 ;+--------------------------------------------------------------------------+
```

```
                        427          .SBTTL  "       L0011 & L0016 GATE ARRAY LOCATION"
                        428 ;+-----------------------------------------------------------------------+
                        429 ;|                                                                       |
                        430 ;|                                                                       |
                        431 ;|                                                                       |
                        432 ;|                                                                       |
                        433 ;|                                                                       |
                        434 ;|                                                                       |
                        435 ;|                                                                       |
                        436 ;|                                                                       |
                        437 ;|                                                                       |
                        438 ;|                                                                       |
                        439 ;|                                                                       |
                        440 ;|                                                                       |
                        441 ;|                                                                       |
                        442 ;|                                                                       |
                        443 ;|                                                                       |
                        444 ;|                                                                       |
                        445 ;|                                                                       |
                        446 ;|                                                                       |
                        447 ;|                                                                       |
                        448 ;|                                                                       |
                        449 ;|                                                                       |
                        450 ;|                                                                       |
                        451 ;|                                                                       |
                        452 ;|                                                                       |
                        453 ;|                                                                       |
                        454 ;|                                                                       |
                        455 ;|                                                                       |
                        456 ;|                                                                       |
                        457 ;|                                                                       |
                        458 ;|                                                                       |
                        459 ;|                                                                       |
                        460 ;|                                                                       |
                        461 ;|                                                                       |
                        462 ;|                                                                       |
                        463 ;|                                                                       |
                        464 ;|                                                                       |
                        465 ;|                                                                       |
                        466 ;|                                                                       |
                        467 ;|                                                                       |
                        468 ;|                                                                       |
                        469 ;|                                                                       |
                        470 ;|                                                                       |
                        471 ;|                                                                       |
                        472 ;|                                                                       |
                        473 ;|                                                                       |
                        474 ;|                                                                       |
                        475 ;|                                                                       |
                        476 ;|                                                                       |
                        477 ;|                                                                       |
                        478 ;|                                                                       |
                        479 ;|                                                                       |
                        480 ;|                                                                       |
                        481 ;|                                                                       |
```



```
0000   482 ;|                                                                                |
0000   483 ;+--------------------------------------------------------------------------------+
```

```
                        0000    485         .SBTTL  ""
                        0000    486         .LIST   MC
                        0000    487         .LIBRARY        /UCOD$0:[VAX750]COMETMAC/
                        0000    488         .LIST   ME
                        0000    489         .NLIST  CND,MC
                        0000                .SBTTL  "EQUATED SYMBOLS
                        0000        ;
                        0000        ; RDM REGISTER DEFINITIONS:
                        0000        ;
                        0000        ; NOTE: THESE REGITER DEFINITIONS ARE OFFSET BY THE LOAD/START ADDRESS OF
                        0000        ;       THE PROGRAM. FOR THE SIMULATOR VERSION THIS IS 400(X) AND FOR THE
                        0000        ;       REAL VERSION IT IS 8400(X).
                        0000        ;
00007400                0000                DCS_DATA_REG    =       ^XF800 + HEAD
00007401                0000                DCS_ADDR_REG_WO =       ^XF801 + HEAD
00007402                0000                DCS_CTRL_REG    =       ^XF802 + HEAD
00007403                0000                DCS_CTRL_FILE   =       ^XF803 + HEAD
00007404                0000                ADDR_MATCH_LO   =       ^XF804 + HEAD
00007405                0000                ADDR_MATCH_HI   =       ^XF805 + HEAD
00007406                0000                STATUS_REG      =       ^XF806 + HEAD
                        0000        ;
00007410                0000                A_REG_BYTE_0    =       ^XF810 + HEAD
00007411                0000                A_REG_BYTE_1    =       ^XF811 + HEAD
00007412                0000                A_REG_BYTE_2    =       ^XF812 + HEAD
00007413                0000                A_REG_BYTE_3    =       ^XF813 + HEAD
00007414                0000                MD_REG_BYTE_0   =       ^XF814 + HEAD
00007415                0000                MD_REG_BYTE_1   =       ^XF815 + HEAD
00007416                0000                MD_REG_BYTE_2   =       ^XF816 + HEAD
00007417                0000                MD_REG_BYTE_3   =       ^XF817 + HEAD
00007418                0000                MH_REG_BYTE_0   =       ^XF818 + HEAD
00007419                0000                MH_REG_BYTE_1   =       ^XF819 + HEAD
0000741A                0000                MH_REG_BYTE_2   =       ^XF81A + HEAD
0000741B                0000                MH_REG_BYTE_3   =       ^XF81B + HEAD
0000741C                0000                CMI_CTRL_REG    =       ^XF81C + HEAD
0000741D                0000                MAINT_REG       =       ^XF81D + HEAD
0000741E                0000                RD_CTRL_REG     =       ^XF81E + HEAD
                        0000        ;
00007420                0000                FRONT_PNL_1     =       ^XF820 + HEAD
00007421                0000                FRONT_PNL_2     =       ^XF821 + HEAD
00007422                0000                CS_ADD_TRAP_LOW =       ^XF822 + HEAD
00007423                0000                CS_ADD_TRAP_HGH =       ^XF823 + HEAD
00007424                0000                CS_ADD_BUFF_LOW =       ^XF824 + HEAD
00007425                0000                CS_ADD_BUFF_HGH =       ^XF825 + HEAD
00007426                0000                BUFF_ADDR_LOW   =       ^XF826 + HEAD
00007427                0000                DCS_ADDR_REG_RO =       ^XF827 + HEAD
                        0000        ;
0000742C                0000                WD_REG_BYTE_0   =       ^XF82C + HEAD
0000742D                0000                WD_REG_BYTE_1   =       ^XF82D + HEAD
0000742E                0000                WD_REG_BYTE_2   =       ^XF82E + HEAD
0000742F                0000                WD_REG_BYTE_3   =       ^XF82F + HEAD
00007430                0000                WH_REG_BYTE_0   =       ^XF830 + HEAD
00007431                0000                WH_REG_BYTE_1   =       ^XF831 + HEAD
00007432                0000                WH_REG_BYTE_2   =       ^XF832 + HEAD
00007433                0000                WH_REG_BYTE_3   =       ^XF833 + HEAD
                        0000        ;
```

```
                        0000        ; RDM REGISTER NAMES AS DEFINED IN HARDWARE SPEC
                        0000        ;
                00007400 0000               DCSDA=DCS_DATA_REG              ;Diagnostic control store data register
                00007401 0000               DCSAD=DCS_ADDR_REG_WO           ;Diagnostic control store address reg
                00007402 0000               DCSCR=DCS_CTRL_REG              ;Diagnostic control store control reg
                00007403 0000               DCSCF=DCS_CTRL_FILE             ;Diagnostic control store control file
                00007404 0000               CSAMR=ADDR_MATCH_LO             ;Control store address match register
                00007406 0000               STATUS=STATUS_REG               ;Status register
                00007410 0000               MAREG=A_REG_BYTE_0              ;Memory address register
                00007414 0000               MDREG=MD_REG_BYTE_0             ;Memory data register
                00007418 0000               MHREG=MH_REG_BYTE_0             ;Memory holding register
                0000741C 0000               CMICTL=CMI_CTRL_REG             ;CMI control
                0000741D 0000               MAIN32=MAINT_REG                ;32 Bit path maintenance control
                0000741E 0000               RDCTRL=RD_CTRL_REG              ;Remote diagnosis control
                00007420 0000               FRONT1=FRONT_PNL_1              ;Front panel readback 1
                00007421 0000               FRONT2=FRONT_PNL_2              ;Front panel readback 2
                00007422 0000               CSTRP=CS_ADD_TRAP_LOW           ;Control store address trap register
                00007424 0000               CSBUF=CS_ADD_BUFF_LOW           ;Control store address buffer
                00007426 0000               BADD1=BUFF_ADDR_LOW             ;Control store address trace readback
                00007427 0000               CLKDCS=DCS_ADDR_REG_RO          ;Clock control & DCS address register
                0000742C 0000               WDREG=WD_REG_BYTE_0             ;WBUS data register
                00007430 0000               WHREG=WH_REG_BYTE_0             ;WBUS holding register
                        0000        ;
                        0000        ; RDM REGISTER BIT DEFINITIONS:
                        0000        ;
                        0000        ; DCS CONTROL REGISTER
                00000001 0000               HALT_ON_MATCH    =      1
                00000002 0000               CLEAR_CTRL_FILE  =      2
                00000004 0000               TRACE_ENABL      =      4
                00000008 0000               PAR_CHK_ENABL    =      8
                00000010 0000               PAR_STOP_ENABL   =      ^X10
                00000020 0000               CLK_CTRL_0       =      ^X20    ; ACTIVE LOW
                00000040 0000               CLK_CTRL_1       =      ^X40    ; ACTIVE LOW
                00000080 0000               MICRO_ADDR_INH   =      ^X80
                        0000            ;
                        0000            ; FOLLOWING IS A FUNCTION TABLE OF THE CLOCK CONTROL BITS
                        0000            ;
                        0000                ;       CLK_CTRL 1,0    FUNCTION
                        0000                ;           11            HALT
                        0000                ;           01            SINGLE MICRO INSTRUCTION
                        0000                ;           10            SINGLE TICK
                        0000                ;           00            RUN
                        0000        ;
                        0000        ; DCS CONTROL FILE      ALL OF THESE BITS ARE ACTIVE LOW
                        0000        ;
                00000001 0000               V_BUS_STROBE     =      1
                00000002 0000               DCS_ADDR_CLEAR   =      2
                00000004 0000               STOP_CLOCK       =      4
                00000008 0000               EN_TRACE_REG     =      8
                00000010 0000               H_FROM_WBUS      =      ^X10
                00000020 0000               WBUS_FROM_D      =      ^X20
                00000040 0000               STROBE_CMI       =      ^X40
                00000080 0000               SA_CLOCK         =      ^X80
                        0000        ;
                        0000        ; CS ADDRESS MATCH REGISTER (HIGH)
```

```
                    0000        ;
00000040            0000                VBUS_SERIAL_IN  =       ^X40    ; ACTIVE LOW
00000080            0000                VBUS_CLOCK      =       ^X80
                    0000        ;
                    0000        ; STATUS REGISTER
                    0000        ;
00000001            0000                VBUS_SERIAL_OUT =       1
00000002            0000                TRAP_OFF        =       2
00000008            0000                CMI_STATUS_0    =       8
00000010            0000                CMI_STATUS_1    =       ^X10
                    0000                ;
                    0000                ; THE CMI STATUS BITS ARE ENCODED AS FOLLOWS:
                    0000                ;
                    0000                ;       00 = NO RESPONSE
                    0000                ;       01 = UNCORRECTABLE ERROR
                    0000                ;       10 = CORRECTABLE ERROR
                    0000                ;       11 = NO ERRORS
00000020            0000                CMI_COMPLETE    =       ^X20
                    0000        ;
00000080            0000                CLOCK_STOPED    =       ^X80
                    0000        ;
                    0000        ; CMI CONTROL REGISTER
                    0000        ;
00000001            0000                CMI_CTRL_CLEAR  =       1
00000008            0000                CMI_GO          =       8
00000020            0000                CMI_WRITE       =       ^X20
00000040            0000                CMI_READ        =       ^X40
00000080            0000                SA_START_STOP   =       ^X80
                    0000        ;
                    0000        ; 32 BIT MAINTENANCE CONTROL REGISTER
                    0000        ;
00000001            0000                MAINT_TRAP_OFF  =       1
00000002            0000                MAINT_STB_INH   =       2
00000004            0000                MAINT_DCS_ENABL =       4
00000008            0000                MAINT_WB_TO_WH  =       8
00000010            0000                MAINT_WD_TO_WB  =       ^X10
00000020            0000                MAINT_CMI_TO_MH =       ^X20
00000040            0000                MAINT_MD_TO_CMI =       ^X40
00000080            0000                MAINT_A_TO_CMI  =       ^X80
                    0000        ;
                    0000        ; RD CONTROL REGISTER
                    0000        ;
00000008            0000                MASTER_HALT_EN  =       8
00000010            0000                VBUS_LOAD       =       ^X10
00000020            0000                TRAP_HALT_EN    =       ^X20
00000040            0000                DC_LOW          =       ^X40
00000080            0000                AC_LOW          =       ^X80
                    0000        ;
                    0000        ; FRONT PANEL REGISTER 1
                    0000        ;
00000001            0000                FP_BOOT_0       =       1
00000002            0000                FP_BOOT_1       =       2
00000004            0000                FP_START_0      =       4
00000008            0000                FP_START_1      =       8
00000010            0000                CPU_RUN         =       ^X10
```

```
00000020  0000                PARITY_ERROR    =       ^X20
          0000        ;
          0000        ; FRONT PANEL REGISTER 2
          0000        ;
00000001  0000                REMOTE          =       1
00000002  0000                FAULT           =       2
00000004  0000                TEST            =       4
          0000        ;
          0000        ;
          0000        ;
00000040  0000                PB_INIT         =       ^X40    ; ACTIVE LOW
00000080  0000                FRONT_PNL_LOCK  =       ^X80
          0000        ;
          0000        ; DCS ADDRESS REGISTER READ ONLY
          0000        ;
00000040  0000                CLK_CTRL_0_RO   =       ^X40    ; ACTIVE HIGH
00000080  0000                CLK_CTRL_1_RO   =       ^X80    ; ACTIVE HIGH
          0000        ;
          0000        ; MISCELLANEOUS EQUATES:
          0000        ;
00000001  0000                BYTE            =       1
00000002  0000                WORD            =       2
00000004  0000                LONG            =       4
0000000A  0000                MICROWORD       =       10
00002C5E  0000                MONITOR_SIZE    =       ^X2C5E
00000400  0000                STACK_SIZE      =       1024
000007A2  0000                M$TEST_SIZE     =       14336 - STACK_SIZE - MONITOR_SIZE
          0000                                              ; MAXIMUM SIZE OF A TEST OVERLAY
000003E2  0000                M$TEST_SIZE_D   = M$TEST_SIZE - 960
          0000                                              ; MAXIMUM SIZE OF TEST OVERLAY WITH
          0000                                              ; 'DEBUG' ENABLED IN
00000001  0000                B               =       BYTE
00000002  0000                W               =       WORD
00000004  0000                L               =       LONG
0000000A  0000                M               =       MICROWORD
00000001  0000                HIGH            =       1
00000000  0000                LOW             =       0
00000000  0000                ONERROR         =       0
00000001  0000                NOERROR         =       1
00000002  0000                ONEQUAL         =       2
00000003  0000                NOTEQUAL        =       3
00000010  0000                MAX_ARGUMENTS   =       16      ; BYTE LENGTH OF LARGEST PSEUDO OP
00000036  0000                DCS_SCRATCH_ADR =       54      ; DCS SCRATCH ADDRESS START
000032FC  0000                ERROR_LOG_ADR   =       ^XB6FC + HEAD
          0000                                              ; START ADDRESS OF MEMORY ERROR LOG
          0000    491         .NLIST  ME
          0000    492         .LIST   MC
          0000    493         .SBTTL  ""
          0000    494         .NLIST  MC      ; SHUT-OFF LISTING FOR FIRST 'NEWTEST' CALL
```

```
                        0020        .SBTTL  "TEST   0B6     TEST BRANCHING ON WBUS DATA
                        0020      3 ;*******************************************************************************
                        0020      4 ;++
                        0020      5 ;
                        0020      6 ; FUNCTIONAL DESCRIPTION:
                        0020      7 ;
                        0020      8 ;       THIS TEST CHECKS ALL THE BUT MICRO ORDERS THAT USE THE WBUS BITS IN
                        0020      9 ;       THE PHB GATE ARRAY.
                        0020     10 ;
                        0020     11 ;
                        0020     12 ; TEST ALGORITHM:
                        0020     13 ;
                        0020     14 ;       1.  SET UP INDEX K FOR THE CMPREGMSK PSEUDO OP
                        0020     15 ;       2.  SET UP INDEX I TO CHANGE THE MICRO CODE
                        0020     16 ;       3.  LOAD THE MICRO CODE FOR THIS LOOP (I)
                        0020     17 ;       4.  SET UP INDEX J TO SELECT TEST PATTERNS
                        0020     18 ;       5.  LOAD A TEST PATTERN INTO WDREG
                        0020     19 ;       6.  EXECUTE DCS PROGRAM
                        0020     20 ;               a.  BRANCH ON THE WBUS DATA
                        0020     21 ;       7.  TEST THE CS ADDRESS FOR THE CORRECT BRANCH
                        0020     22 ;       8.  IF NO ERROR INCREMENT INDEX K
                        0020     23 ;       9.  GO TO STEP 5 UNTIL J IS EXHAUSTED
                        0020     24 ;      10.  GO TO STEP 3 UNTIL I IS EXHAUSTED
                        0020     25 ;
                        0020     26 ;
                        0020     27 ; TEST PATTERNS:
                        0020     28 ;
                        0020     29 ;       ITERATION (K)   MICRO ORDER     TEST PATTERN    CS ADDRESS
                        0020     30 ;             1             WBUS1TO0        8000002A        1802
                        0020     31 ;             2                "            40000015        1801
                        0020     32 ;             3                "            C0000033        1803
                        0020     33 ;             4                "            0000000F        1803
                        0020     34 ;             5                "            00000000        1800
                        0020     35 ;             6          WBUS1TO0.NE.0      8000002A        1801
                        0020     36 ;             7                "            40000015        1801
                        0020     37 ;             8                "            C0000033        1801
                        0020     38 ;             9                "            0000000F        1801
                        0020     39 ;             A                "            00000000        1800
                        0020     40 ;             B             WBUS5TO0        8000002A        182A
                        0020     41 ;             C                "            40000015        1815
                        0020     42 ;             D                "            C0000033        1833
                        0020     43 ;             E                "            0000000F        180F
                        0020     44 ;             F                "            00000000        1800
                        0020     45 ;            10             WBUS0           8000002A        1800
                        0020     46 ;            11                "            40000015        1801
                        0020     47 ;            12                "            C0000033        1801
                        0020     48 ;            13                "            0000000F        1801
                        0020     49 ;            14                "            00000000        1800
                        0020     50 ;            15          WBUS31TO30         8000002A        1802
                        0020     51 ;            16                "            40000015        1801
                        0020     52 ;            17                "            C0000033        1803
                        0020     53 ;            18                "            0000000F        1800
                        0020     54 ;            19                "            00000000        1800
                        0020     55 ;
                        0020     56 ;
```

```
                                  0020   ; LOGIC DESCRIPTION:
                                  0020   ;
                                  0020   ;       THIS TEST CHECKS THE FIVE MICRO ORDERS THAT BUT ON THE WBUS DATA GATED
                                  0020   ;       THROUGH THE PHB GATE ARRAY.  ANY ERRORS WOULD PROBABLY BE IN THE GATE
                                  0020   ;       ARRAY OR THE PHB GOOD SAMARITAN ROM.
                                  0020   ;
                                  0020   ;
                                  0020   ; ASSUMPTIONS:
                                  0020   ;
                                  0020   ;       THIS TEST ASSUMES THE WBUS IS OPERATIONAL.
                                  0020   ;
                                  0020   ;
                                  0020   ; ERROR DESCRIPTION:
                                  0020   ;
                                  0020   ;       EXPECTED CS ADDRESS
                                  0020   ;       RECEIVED CS ADDRESS
                                  0020   ;       LOOP COUNT I
                                  0020   ;       LOOP COUNT J
                                  0020   ;       LOOP COUNT K
                                  0020   ;
                                  0020   ;--
                                  0020   ;*******************************************************************************
                                  0020
                                  0020
                                  0020           INITIALIZE                                  ;INIT THE CPU
                                  0022           LDINDEX         K,1                         ;SET K TO 0
                                  0029           LOOP            I,1,5                       ;CREATE A LOOP OF 5
                                  0030           LOADDCS         1$,I,01,1                   ;LOAD MICRO CODE FOR THIS LOOP
                                  0037           LOOP            J,1,5                       ;CREATE A SECOND LOOP OF 5
                                  003E           LOADREG         WDREG,2$,J,L                ;LOAD TEST PATTERN INTO WDREG
                                  0046           ERRLOOP                                     ;ERROR LOOP POINT
                                  0048           FETCH           0                           ;START DCS PROGRAM
                                  004D           BRSTCLK         2                           ;END DCS PROGRAM
                                  0051           CMPREGMSK       CSTRP,3$,K,4$,,W,           ;TEST CS ADDRESS
                                  005D           IFERROR         ,<PHB>,                     ;INCORRECT BRANCH
                                  0063           INCINDEX        K                           ;INCREMENT INDEX K
                                  0066           ENDLOOP         J                           ;END TEST LOOP J
                                  0069           ENDLOOP         I                           ;END TEST LOOP I
                                  006C           SKIP                                        ;GO TO NEXT TEST
                                  0073
                                  0073
                                  0073   BEGIN_TEST_DATA
                                  0073
                                  0073       ;-----------------------------------------------------------------------
                                  0073
                                  0073   1$:                                                 ;MICRO CODE FOR EACH LOOP
U 01, 5700,4800,0364,2700,0470,1800  0073   ;% 01   WB_RDM,BUT/WBUS1TO0,NEXT/1800
U 01, 5700,0800,0364,3B00,0470,1800  007E   ;% 01   WB_RDM,BUT/WBUS1TO0.NE.0,NEXT/1800
U 01, 5700,0800,0364,2300,0470,1800  0089   ;% 01   WB_RDM,BUT/WBUS5TO0,NEXT/1800
U 01, 5700,4800,0364,2B00,0470,1800  0094   ;% 01   WB_RDM,BUT/WBUS0,NEXT/1800
U 01, 5700,4800,0364,6F00,0470,1800  009F   ;% 01   WB_RDM,BUT/WBUS31TO30,NEXT/1800
                          8000002A  00AA   2$:     .LONG   ^X8000002A                          ;TEST PATTERN TABLE
                          40000015  00AE           .LONG   ^X40000015
                          C0000033  00B2           .LONG   ^XC0000033
                          0000000F  00B6           .LONG   ^X0000000F
```

```
                    00000000  00BA   125          .LONG   ^X00000000
                        1802  00BE   126 3$:      .WORD   ^X1802                          ;RESULTS TABLE
                        1801  00C0   127          .WORD   ^X1801
                        1803  00C2   128          .WORD   ^X1803
                        1803  00C4   129          .WORD   ^X1803
                        1800  00C6   130          .WORD   ^X1800
                        1801  00C8   131          .WORD   ^X1801
                        1801  00CA   132          .WORD   ^X1801
                        1801  00CC   133          .WORD   ^X1801
                        1801  00CE   134          .WORD   ^X1801
                        1800  00D0   135          .WORD   ^X1800
                        182A  00D2   136          .WORD   ^X182A
                        1815  00D4   137          .WORD   ^X1815
                        1833  00D6   138          .WORD   ^X1833
                        180F  00D8   139          .WORD   ^X180F
                        1800  00DA   140          .WORD   ^X1800
                        1800  00DC   141          .WORD   ^X1800
                        1801  00DE   142          .WORD   ^X1801
                        1801  00E0   143          .WORD   ^X1801
                        1801  00E2   144          .WORD   ^X1801
                        1800  00E4   145          .WORD   ^X1800
                        1802  00E6   146          .WORD   ^X1802
                        1801  00E8   147          .WORD   ^X1801
                        1803  00EA   148          .WORD   ^X1803
                        1800  00EC   149          .WORD   ^X1800
                        1800  00EE   150          .WORD   ^X1800
                        3FFF  00F0   151 4$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                              00F2   152
                              00F2   153
                              00F2   154 BEGIN_CPU_DATA
                              0002   155
                              0002   156 DCS_DATA
                              0001   157
U 00, 7700,C800,0364,0300,0470,1801   0001   158 ;% 00    NOP
U 01, 5700,4800,0364,2700,0470,1800   000C   162 ;% 01    WB_RDM,BUT/WBUS1TO0,NEXT/1800           ;BUT ON WBUS DATA
U 02, 7300,4800,0364,0300,0470,1803   0017   166 ;% 02    NOP,STOP.CLOCK                          ;STOP THE CPU CLOCK
                              0022   170
                              0022   171 END_CPU_DATA
                              00F2   172
                              00F2   173
                              00F2   174 END_TEST_DATA
                              00F2
                              00F2       ;-----------------------------------------------------------------------------
                              00F2   175 ENDTEST
```

```
                        002B        .SBTTL  "TEST   0B7     TEST STATUS FLAGS WITH WCTRL FUNCTIONS
                        002B    177 ;*******************************************************************************
                        002B    178 ;++
                        002B    179 ;
                        002B    180 ; FUNCTIONAL DESCRIPTION:
                        002B    181 ;
                        002B    182 ;       THIS TEST WRITES AND READS THE CM.TP.FPD.FLAGS IN THE PHB GATE ARRAY.
                        002B    183 ;
                        002B    184 ;
                        002B    185 ; TEST ALGORITHM:
                        002B    186 ;
                        002B    187 ;       1.  CREATE A TEST LOOP OF 4
                        002B    188 ;       2.  LOAD A TEST PATTERN INTO THE WDREG
                        002B    189 ;       3.  EXECUTE DCS PROGRAM
                        002B    190 ;               a.  WRITE THE TEST PATTERN TO CM.TP.FPD
                        002B    191 ;               b.  WRITE THE TEST PATTERN TO FLAGS
                        002B    192 ;               c.  READ CM.TP.FPD.FLAGS TO THE RDM WHREG
                        002B    193 ;       4.  TEST THE WHREG FOR THE CORRECT DATA
                        002B    194 ;       5.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST PATTERNS
                        002B    195 ;
                        002B    196 ;
                        002B    197 ; TEST PATTERNS:
                        002B    198 ;
                        002B    199 ;       ITERATION       TEST PATTERN
                        002B    200 ;           1             8800002A
                        002B    201 ;           2             40000015
                        002B    202 ;           3             00000033
                        002B    203 ;           4             0000000F
                        002B    204 ;
                        002B    205 ;
                        002B    206 ; LOGIC DESCRIPTION:
                        002B    207 ;
                        002B    208 ;       THIS TEST WRITES THE PSL CM, TP, FPD BITS AS WELL AS THE STAUS FLAGS
                        002B    209 ;       WHICH ARE ALL LOCATED IN THE PHB GATE ARRAY.  THE DATA IS THEN READ
                        002B    210 ;       BACK TO THE RDM USING THE WCTRL/CM.TP.FPD.FLAGS MICRO ORDER.  ANY
                        002B    211 ;       ERRORS SHOULD BE IN THE PHB ARRAY OR THE PHB GOOD SAMARITAN ROM.
                        002B    212 ;
                        002B    213 ;
                        002B    214 ; ASSUMPTIONS:
                        002B    215 ;
                        002B    216 ;       THIS TEST ASSUMES THE WBUS IS OPERATIONAL.
                        002B    217 ;
                        002B    218 ;
                        002B    219 ; ERROR DESCRIPTION:
                        002B    220 ;
                        002B    221 ;       EXPECTED TEST PATTERN
                        002B    222 ;       RECEIVED TEST PATTERN
                        002B    223 ;       LOOP COUNT I
                        002B    224 ;
                        002B    225 ;--
                        002B    226 ;*******************************************************************************
                        002B    227
                        002B    228
                        002B    229
                        002B    230         INITIALIZE                                  ;INIT THE CPU
```

```
                                002D          LOOP            I,1,4                   ;CREATE A TEST LOOP OF 4
                                0034          LOADREG         WDREG,1$,I,L            ;PUT TEST PATTERN IN WDREG
                                003C          ERRLOOP                                 ;ERROR LOOP POINT
                                003E          FETCH           0                       ;START DCS PROGRAM
                                0043          BRSTCLK         4                       ;END DCS PROGRAM
                                0047          CMPREGMSK       WHREG,1$,I,2$,,L,       ;TEST RETURNED TEST PATTERN
                                0053          IFERROR         ,<PHB>,                 ;PATTERN INCORRECT
                                0059          ENDLOOP         I                       ;END TEST LOOP
                                005C          SKIP                                    ;GO TO NEXT TEST
                                0063
                                0063
                                0063    BEGIN_TEST_DATA
                                0063
                                0063    ;-----------------------------------------------------------------------
                                0063
                      8800002A  0063    1$:     .LONG   ^X8800002A                      ;TEST PATTERN TABLE
                      40000015  0067            .LONG   ^X40000015
                      00000033  006B            .LONG   ^X00000033
                      0000000F  006F            .LONG   ^X0000000F
                      C800003F  0073    2$:     .LONG   ^XC800003F                      ;MASK FOR CMPREGMSK PSEUDO OP
                                0077
                                0077
                                0077
                                0077    BEGIN_CPU_DATA
                                0002
                                0002    DCS_DATA
                                0001
U 00, 7700,C800,0364,0300,0470,1801  0001    ;% 00   NOP
U 01, 5700,4800,0364,0300,0070,1802  000C    ;% 01   CCPSL/PSL_WB.CCBR_ALUS,WB_RDM           ;SET CM, TP, FPD BITS
U 02, 5700,C800,0364,0300,3070,1803  0017    ;% 02   WCTRL/FLAGS_WB,WB_RDM                  ;SET STATUS FLAGS
U 03, 6700,C800,0364,0300,3870,1804  0022    ;% 03   RDM_WB,WCTRL/CM.TP.FPD.FLAGS            ;READ BACK TO RDM
U 04, 7300,4800,0364,0300,0470,1805  002D    ;% 04   NOP,STOP.CLOCK                         ;STOP THE CPU CLOCK
                                0038
                                0038    END_CPU_DATA
                                0077
                                0077
                                0077    END_TEST_DATA
                                0077
                                0077    ;-----------------------------------------------------------------------
                                0077    ENDTEST
```

```
                        002C        .SBTTL  "TEST   0B8     TEST STEP COUNTER USING WCTRL FUNCTIONS
                        002C    283 ;*******************************************************************************
                        002C    284 ;++
                        002C    285 ;
                        002C    286 ; FUNCTIONAL DESCRIPTION:
                        002C    287 ;
                        002C    288 ;       THIS TEST WILL CHECK THE CM, TP, FPD, F5, AND STEP COUNTER BITS IN
                        002C    289 ;       THE PHB GATE ARRAY USING THE WCTRL FUNCTIONS.
                        002C    290 ;
                        002C    291 ;
                        002C    292 ; TEST ALGORITHM:
                        002C    293 ;
                        002C    294 ;       1.  CREATE A TEST LOOP OF 4
                        002C    295 ;       2.  LOAD A TEST PATTERN INTO THE WDREG
                        002C    296 ;       3.  EXECUTE DCS PROGRAM
                        002C    297 ;               a.  SET UP CM, TP, AND FPD BITS
                        002C    298 ;               b.  SET UP STEP COUNTER AND FLAG5
                        002C    299 ;               c.  READ THE DATA BACK TO THE RDM
                        002C    300 ;       4.  TEST THE RETURNED DATA
                        002C    301 ;       5.  IF NO ERROR GO TO STEP 2 UNTIL THE TEST LOOP IS EXHAUSTED
                        002C    302 ;
                        002C    303 ;
                        002C    304 ; TEST PATTERNS:
                        002C    305 ;
                        002C    306 ;       ITERATION       TEST PATTERN
                        002C    307 ;           1             8800002A
                        002C    308 ;           2             40000035
                        002C    309 ;           3             00000033
                        002C    310 ;           4             0000002F
                        002C    311 ;
                        002C    312 ;
                        002C    313 ; LOGIC DESCRIPTION:
                        002C    314 ;
                        002C    315 ;       THIS TEST CHECKS THE CM, TP, FPD, FLAG5, AND STEP COUNTER BITS
                        002C    316 ;       HOUSED IN THE PHB GATE ARRAY.  ANY FAILURES SHOULD BE CONFINED TO
                        002C    317 ;       THE ARRAY OR THE PHB GOOD SAMARITAN ROM.
                        002C    318 ;
                        002C    319 ;
                        002C    320 ; ASSUMPTIONS:
                        002C    321 ;
                        002C    322 ;       THIS TEST ASSUMES THE WBUS IS OPERATIONAL
                        002C    323 ;
                        002C    324 ;
                        002C    325 ; ERROR DESCRIPTION:
                        002C    326 ;
                        002C    327 ;       EXPECTED TEST DATA
                        002C    328 ;       RECEIVED TEST DATA
                        002C    329 ;       LOOP COUNT I
                        002C    330 ;
                        002C    331 ;--
                        002C    332 ;*******************************************************************************
                        002C    333
                        002C    334
                        002C    335
                        002C    336         INITIALIZE                                  ;INIT THE CPU
```

```
                                002E          LOOP            I,1,4                   ;CREATE A TEST LOOP OF 4
                                0035          LOADREG         WDREG,1$,I,L            ;PUT TEST PATTERN INTO WDREG
                                003D          ERRLOOP                                 ;ERROR LOOP POINT
                                003F          FETCH           0                       ;START DCS PROGRAM
                                0044          BRSTCLK         4                       ;END DCS PROGRAM
                                0048          CMPREGMSK       WHREG,1$,I,2$,,L,       ;CHECK RETURNED TEST PATTERN
                                0054          IFERROR         ,<PHB>,                 ;INCORRECT TEST PATTERN
                                005A          ENDLOOP         I                       ;END TEST LOOP
                                005D          SKIP                                    ;GO TO NEXT TEST
                                0064
                                0064
                                0064  BEGIN_TEST_DATA
                                0064
                                0064      ;-----------------------------------------------------------------------
                                0064
                      8800002A  0064  1$:     .LONG   ^X8800002A                      ;TEST PATTERN TABLE
                      40000035  0068          .LONG   ^X40000035
                      00000033  006C          .LONG   ^X00000033
                      0000002F  0070          .LONG   ^X0000002F
                      C800003F  0074  2$:     .LONG   ^XC800003F                      ;MASK FOR CMPREGMSK PSEUDO OP
                                0078
                                0078
                                0078
                                0078  BEGIN_CPU_DATA
                                0002
                                0002  DCS_DATA
                                0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 5700,4800,0364,0300,0070,1802  000C  ;% 01   CCPSL/PSL_WB.CCBR_ALUS,WB_RDM           ;LOAD CM, TP, AND FPD BITS
U 02, 5700,4680,0364,0300,1070,1803  0017  ;% 02   WCTRL/STEPC_WB,WB_RDM,MISC/SET.STACKFLG ;LOAD STEP COUNTER AND FLAG5
U 03, 6700,4800,0364,0300,1870,1804  0022  ;% 03   RDM_WB,WCTRL/CM.TP.FPD.F5.STEPC         ;READ THE WHOLE MESS BACK
U 04, 7300,4800,0364,0300,0470,1805  002D  ;% 04   NOP,STOP.CLOCK                          ;STOP THE CPU CLOCK
                                0038
                                0038  END_CPU_DATA
                                0078
                                0078
                                0078  END_TEST_DATA
                                0078
                                0078      ;-----------------------------------------------------------------------
                                0078  ENDTEST
```

```
                        0022        .SBTTL  "TEST   0B9     BRANCH ON STATUS FLAGS TEST 1
                        0022    389 ;*****************************************************************************
                        0022    390 ;++
                        0022    391 ;
                        0022    392 ; FUNCTIONAL DESCRIPTION:
                        0022    393 ;
                        0022    394 ;       THIS TEST CHECKS THE ABILITY OF CPU TO BRANCH ON THE STATUS FLAGS.
                        0022    395 ;
                        0022    396 ;
                        0022    397 ; TEST ALGORITHM:
                        0022    398 ;
                        0022    399 ;       1.  CREATE AN INDEX K TO SELECT MICRO CODE AND RESULTS
                        0022    400 ;       2.  CREATE A LOOP OF 6 TO SELECT BUT MICRO CODE
                        0022    401 ;       3.  LOAD BUT MICRO ORDER
                        0022    402 ;       4.  CREATE A SECOND LOOP OF J TO HAVE MICRO CODE EXECUTE TWICE
                        0022    403 ;       5.  LOAD MISC MICRO ORDER
                        0022    404 ;       6.  EXECUTE DCS PROGRAM
                        0022    405 ;               a.  CONDITION FLAGS
                        0022    406 ;               b.  BRANCH ON FLAG STATE
                        0022    407 ;       7.  TEST CS ADDRESS FOR CORRECT BRANCH
                        0022    408 ;       8.  IF NO ERROR INCREMENT K BY 1
                        0022    409 ;       9.  GO TO STEP 5 UNTIL J IS EXHAUSTED
                        0022    410 ;      10.  GO TO STEP 3 UNTIL I IS EXHAUSTED
                        0022    411 ;
                        0022    412 ;
                        0022    413 ; TEST PATTERNS:
                        0022    414 ;
                        0022    415 ;       INDEX (K)                   BUT             MISC            CS ADDRESS
                        0022    416 ;
                        0022    417 ;           1                   FLAG0           SET.FLAG0           1801
                        0022    418 ;           2                   FLAG0           CLR.FLAG0           1800
                        0022    419 ;           3                   FLAG1           SET.FLAG1           1802
                        0022    420 ;           4                   FLAG1           CLR.FLAG1           1800
                        0022    421 ;           5                   FLAG2           SET.FLAG2           1804
                        0022    422 ;           6                   FLAG2           CLR.FLAG2           1800
                        0022    423 ;           7                   FLAG3           SET.FLAG3           1801
                        0022    424 ;           8                   FLAG3           CLR.FLAG3           1800
                        0022    425 ;           9                   MM.NOINT        SET.MMNOINT         1801
                        0022    426 ;           A                   MM.NOINT        CLR.MMNOINT         1800
                        0022    427 ;           B                   STACKFLG        SET.STACKFLG        1801
                        0022    428 ;           C                   STACKFLG        CLR.STACKFLG        1800
                        0022    429 ;
                        0022    430 ;
                        0022    431 ; LOGIC DESCRIPTION:
                        0022    432 ;
                        0022    433 ;       THIS TEST CHEKS THAT THE PHB GATE ARRAY CAN BRANCH THE CS ADDRESS
                        0022    434 ;       LINES BASED ON THE CONDITION OF THE STATUS FLAGS.  ERRORS SHOULD
                        0022    435 ;       EITHER BE CAUSED BY PHB OR THE PHB GOOD SAMARATIN ROM.
                        0022    436 ;
                        0022    437 ;
                        0022    438 ; ASSUMPTIONS:
                        0022    439 ;
                        0022    440 ;       THIS TEST ASSUMES STATUS FLAGS HAVE BEEN TESTED FROM THE WBUS
                        0022    441 ;
                        0022    442 ;
```

```
                        0022   443 ; ERROR DESCRIPTION:
                        0022   444 ;
                        0022   445 ;       EXPECTED CS ADDRESS
                        0022   446 ;       RECEIVED CS ADDRESS
                        0022   447 ;       LOOP COUNT I
                        0022   448 ;       LOOP COUNT J
                        0022   449 ;       INDEX K VALUE
                        0022   450 ;
                        0022   451 ;--
                        0022   452 ;*******************************************************************************
                        0022   453
                        0022   454
                        0022   455         INITIALIZE                              ;INIT THE CPU
                        0024   456         LDINDEX         K,1                     ;CREATE AN INDEX K
                        002B   457         LOOP            I,1,6                   ;CREATE A TEST LOOP OF 6
                        0032   458         LOADDCS         1$,I,02,1               ;LOAD BUT MICRO WORD
                        0039   459         LOOP            J,1,2                   ;CREATE A SECOND TEST LOOP OF 2
                        0040   460         LOADDCS         2$,K,01,1               ;LOAD MISC MICRO WORD
                        0047   461         ERRLOOP                                 ;ERROR LOOP POINT
                        0049   462         FETCH           0                       ;START DCS PROGRAM
                        004E   463         BRSTCLK         3                       ;END DCS PROGRAM
                        0052   464         CMPREGMSK       CSTRP,3$,K,4$,,W,       ;TEST CS ADDRESS
                        005E   465         IFERROR         ,<PHB>,                 ;ADDRESS INCORRECT
                        0064   466         INCINDEX        K                       ;ADD ONE TO INDEX K
                        0067   467         ENDLOOP         J                       ;END TEST LOOP J
                        006A   468         ENDLOOP         I                       ;END TEST LOOP I
                        006D   469         SKIP                                    ;GO TO NEXT TEST
                        0074   470
                        0074   471
                        0074   472 BEGIN_TEST_DATA
                        0074
                        0074       ;-------------------------------------------------------------------------
                        0074   473
                        0074   474 1$:                                            ;BRANCH MICRO CODE
U 02. 7700,0800,0364,4300,0470,1800   0074   475 ;% 02   BUT/FLAG0,NEXT/1800
U 02. 7700,4800,0364,5F00,0470,1800   007F   479 ;% 02   BUT/FLAG1,NEXT/1800
U 02. 7700,4800,0364,4B00,0470,1800   008A   483 ;% 02   BUT/FLAG2,NEXT/1800
U 02. 7700,0800,0364,4F00,0470,1800   0095   487 ;% 02   BUT/FLAG3,NEXT/1800
U 02. 7700,4800,0364,4700,0470,1800   00A0   491 ;% 02   BUT/MM.NOINT,NEXT/1800
U 02. 7700,0800,0364,6B00,0470,1800   00AB   495 ;% 02   BUT/STACKFLG,NEXT/1800
                                      00B6   499 2$:                             ;MISC MICRO CODE
U 01. 7700,C400,0364,0300,0470,1802   00B6   500 ;% 01   MISC/SET.FLAG0
U 01. 7700,4000,0364,0300,0470,1802   00C1   504 ;% 01   MISC/CLR.FLAG0
U 01. 7700,4480,0364,0300,0470,1802   00CC   508 ;% 01   MISC/SET.FLAG1
U 01. 7700,C080,0364,0300,0470,1802   00D7   512 ;% 01   MISC/CLR.FLAG1
U 01. 7700,4500,0364,0300,0470,1802   00E2   516 ;% 01   MISC/SET.FLAG2
U 01. 7700,C100,0364,0300,0470,1802   00ED   520 ;% 01   MISC/CLR.FLAG2
U 01. 7700,C580,0364,0300,0470,1802   00F8   524 ;% 01   MISC/SET.FLAG3
U 01. 7700,4180,0364,0300,0470,1802   0103   528 ;% 01   MISC/CLR.FLAG3
U 01. 7700,4600,0364,0300,0470,1802   010E   532 ;% 01   MISC/SET.MMNOINT
U 01. 7700,C200,0364,0300,0470,1802   0119   536 ;% 01   MISC/CLR.MMNOINT
U 01. 7700,C680,0364,0300,0470,1802   0124   540 ;% 01   MISC/SET.STACKFLG
U 01. 7700,4280,0364,0300,0470,1802   012F   544 ;% 01   MISC/CLR.STACKFLG
                                1801   013A   548 3$:     .WORD   ^X1801          ;RESULTS TABLE
                                1800   013C   549         .WORD   ^X1800
```

```
                        1802  013E   550          .WORD   ^X1802
                        1800  0140   551          .WORD   ^X1800
                        1804  0142   552          .WORD   ^X1804
                        1800  0144   553          .WORD   ^X1800
                        1801  0146   554          .WORD   ^X1801
                        1800  0148   555          .WORD   ^X1800
                        1801  014A   556          .WORD   ^X1801
                        1800  014C   557          .WORD   ^X1800
                        1801  014E   558          .WORD   ^X1801
                        1800  0150   559          .WORD   ^X1800
                        3FFF  0152   560 4$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                              0154   561
                              0154   562
                              0154   563
                              0154   564 BEGIN_CPU_DATA
                              0002   565
                              0002   566 DCS_DATA
                              0001   567
U 00, 7700,C800,0364,0300,0470,1801   0001   568 ;% 00    NOP
U 01, 7700,C400,0364,0300,0470,1802   000C   572 ;% 01    MISC/SET.FLAG0                  ;SET-UP FLAG FOR BRANCH
U 02, 7700,0800,0364,4300,0470,1800   0017   576 ;% 02    BUT/FLAG0,NEXT/1800             ;BRANCH ON FLAG CONDITION
U 03, 7300,C800,0364,0300,0470,1804   0022   580 ;% 03    NOP,STOP.CLOCK                  ;STOP THE CPU CLOCK
                              002D   584
                              002D   585 END_CPU_DATA
                              0154   586
                              0154   587
                              0154   588 END_TEST_DATA
                              0154
                              0154       ;-----------------------------------------------------------------------------
                              0154   589 ENDTEST
```

```
                        0022        .SBTTL  "TEST   0BA     BRANCH ON STATUS FLAGS TEST 2
                        0022    591 ;*****************************************************************************
                        0022    592 ;++
                        0022    593 ;
                        0022    594 ; FUNCTIONAL DESCRIPTION:
                        0022    595 ;
                        0022    596 ;       THIS TEST CHECKS CPUS ABILITY TO BRANCH ON THE STATUS FLAGS.
                        0022    597 ;
                        0022    598 ;
                        0022    599 ; TEST ALGORITHM:
                        0022    600 ;
                        0022    601 ;       1.  CREATE A TEST LOOP OF 9
                        0022    602 ;       2.  LOAD BUT MICRO ORDER INDEXED BY LOOP I
                        0022    603 ;       3.  LOAD DATA FOR FLAGS INTO WDREG
                        0022    604 ;       4.  EXECUTE DCS PROGRAM
                        0022    605 ;               a.  LOAD FLAGS WITH TEST PATTERN
                        0022    606 ;               b.  PERFORM BRANCH ON FLAG CONDITIONS
                        0022    607 ;       5.  TEST CS ADDRESS
                        0022    608 ;       6.  IF NO ERROR GO TO STEP 2 UNTIL LOOP IS EXHAUSTED
                        0022    609 ;
                        0022    610 ;
                        0022    611 ; TEST PATTERNS:
                        0022    612 ;
                        0022    613 ;       ITERATION       FLAGS           CS ADDRESS
                        0022    614 ;           1             5               1805
                        0022    615 ;           2             2               1802
                        0022    616 ;           3             1               1801
                        0022    617 ;           4             1               1801
                        0022    618 ;           5             2               1802
                        0022    619 ;           6             2               1802
                        0022    620 ;           7             4               1801
                        0022    621 ;           8             8               1801
                        0022    622 ;           9             C               1800
                        0022    623 ;
                        0022    624 ;
                        0022    625 ; LOGIC DESCRIPTION:
                        0022    626 ;
                        0022    627 ;       THIS TEST INSURES THAT THE PHB GATE ARRAY CAN BRANCH THE CS ADDRESS
                        0022    628 ;       LINES ACCORDING TO THE STATE OF THE STATUS FLAGS.  ERRORS SHOULD BE
                        0022    629 ;       CAUSED EITHER BY PHB OR THE PHB GOOD SAMARATIN ROM.
                        0022    630 ;
                        0022    631 ;
                        0022    632 ; ASSUMPTIONS:
                        0022    633 ;
                        0022    634 ;       THIS TEST ASSUMES THE STATUS FLAGS HAVE BEEN TESTED FROM THE WBUS.
                        0022    635 ;
                        0022    636 ;
                        0022    637 ; ERROR DESCRIPTION:
                        0022    638 ;
                        0022    639 ;       EXPECTED CS ADDRESS
                        0022    640 ;       RECEIVED CS ADDRESS
                        0022    641 ;       LOOP COUNT I
                        0022    642 ;
                        0022    643 ;--
                        0022    644 ;*****************************************************************************
```

```
                                0022   645
                                0022   646
                                0022   647          INITIALIZE                                  ;INIT THE CPU
                                0024   648          LOOP            I,1,9                       ;CREATE A TEST LOOP OF 9
                                002B   649          LOADDCS         1$,I,02,1                   ;LOAD MICRO CODE FOR THIS LOOP
                                0032   650          LOADREG         WDREG,2$,I,L                ;LOAD A TEST PATTERN INTO WDREG
                                003A   651          ERRLOOP                                     ;ERROR LOOP POINT
                                003C   652          FETCH           0                           ;START THE DCS PROGRAM
                                0041   653          BRSTCLK         3                           ;END DCS PROGRAM
                                0045   654          CMPREGMSK       CSTRP,3$,I,4$,,W,           ;TEST CS ADDRESS
                                0051   655          IFERROR         ,<PHB>,                     ;INCORRECT BRANCH
                                0057   656          ENDLOOP         I                           ;END TEST LOOP
                                005A   657          SKIP                                        ;GO TO NEXT TEST
                                0061   658
                                0061   659
                                0061   660 BEGIN_TEST_DATA
                                0061   661      ;-------------------------------------------------------------------------
                                0061   662 1$:                                                  ;MICRO CODE FOR EACH LOOP
U 02, 7700,0800,0364,5B00,0470,1800   0061   663 ;% 02   BUT/FLAG2T00,NEXT/1800
U 02, 7700,0800,0364,5B00,0470,1800   006C   667 ;% 02   BUT/FLAG2T00,NEXT/1800
U 02, 7700,0800,0364,5B00,0470,1800   0077   671 ;% 02   BUT/FLAG2T00,NEXT/1800
U 02, 7700,4800,0364,5300,0470,1800   0082   675 ;% 02   BUT/FLAG1T00,NEXT/1800
U 02, 7700,4800,0364,5300,0470,1800   008D   679 ;% 02   BUT/FLAG1T00,NEXT/1800
U 02, 7700,0800,0364,5700,0470,1800   0098   683 ;% 02   BUT/F1.XOR23,NEXT/1800
U 02, 7700,0800,0364,5700,0470,1800   00A3   687 ;% 02   BUT/F1.XOR23,NEXT/1800
U 02, 7700,0800,0364,5700,0470,1800   00AE   691 ;% 02   BUT/F1.XOR23,NEXT/1800
U 02, 7700,0800,0364,5700,0470,1800   00B9   695 ;% 02   BUT/F1.XOR23,NEXT/1800
                           00000005   00C4   699 2$:      .LONG   ^X00000005                  ;TEST PATTERN TABLE
                           00000002   00C8   700          .LONG   ^X00000002
                           00000001   00CC   701          .LONG   ^X00000001
                           00000001   00D0   702          .LONG   ^X00000001
                           00000002   00D4   703          .LONG   ^X00000002
                           00000002   00D8   704          .LONG   ^X00000002
                           00000004   00DC   705          .LONG   ^X00000004
                           00000008   00E0   706          .LONG   ^X00000008
                           0000000C   00E4   707          .LONG   ^X0000000C
                               1805   00E8   708 3$:      .WORD   ^X1805                      ;RESULTS TABLE
                               1802   00EA   709          .WORD   ^X1802
                               1801   00EC   710          .WORD   ^X1801
                               1801   00EE   711          .WORD   ^X1801
                               1802   00F0   712          .WORD   ^X1802
                               1802   00F2   713          .WORD   ^X1802
                               1801   00F4   714          .WORD   ^X1801
                               1801   00F6   715          .WORD   ^X1801
                               1800   00F8   716          .WORD   ^X1800
                               3FFF   00FA   717 4$:      .WORD   ^X3FFF                      ;MASK FOR CMPREGMSK PSEUDO OP
                                      00FC   718
                                      00FC   719
                                      00FC   720 BEGIN_CPU_DATA
                                      0002   721
                                      0002   722 DCS_DATA
                                      0001   723
U 00, 7700,C800,0364,0300,0470,1801   0001   724 ;% 00   NOP
```

```
U 01, 5700,4800,0364,0300,3070,1802  000C   728 ;% 01   WCTRL/FLAGS_WB,WB_RDM                    ;SET-UP FLAGS FOR BRANCH
U 02, 7700,0800,0364,5B00,0470,1800  0017   732 ;% 02   BUT/FLAG2TOO,NEXT/1800                  ;BRANCH ON FLAGS
U 03, 7300,C800,0364,0300,0470,1804  0022   736 ;% 03   NOP,STOP.CLOCK                          ;STOP THE CLOCK
                                     002D   740
                                     002D   741 END_CPU_DATA
                                     00FC   742
                                     00FC   743
                                     00FC   744 END_TEST_DATA
                                     00FC
                                     00FC       ;-------------------------------------------------------------------------
                                     00FC   745 ENDTEST
```

```
0025          .SBTTL  "TEST   0BB     TEST MISC STEP COUNTER FUNCTIONS
0025    747 ;*****************************************************************************
0025    748 ;++
0025    749 ;
0025    750 ; FUNCTIONAL DESCRIPTION:
0025    751 ;
0025    752 ;       THIS TEST CHECKS THAT THE STEP COUNTER CAN BE SET ACCORDING TO
0025    753 ;       VARIOUS MISC MICRO ORDERS.
0025    754 ;
0025    755 ;
0025    756 ; TEST ALGORITHM:
0025    757 ;
0025    758 ;       1.  CREATE A LOOP OF 4
0025    759 ;       2.  LOAD MISC MICRO ORDER FOR THIS LOOP
0025    760 ;       3.  EXECUTE DCS PROGRAM
0025    761 ;               a.  SET STEP COUNTER ACCORDING TO MICRO ORDER
0025    762 ;               b.  READ STEP COUNTER BACK TO RDM
0025    763 ;       4.  TEST VALUE IN STEP COUNTER
0025    764 ;       5.  IF NO ERROR GO TO STEP 2 UNTIL LOOP IS EXHAUSTED
0025    765 ;
0025    766 ;
0025    767 ; TEST PATTERNS:
0025    768 ;
0025    769 ;       ITERATION       MISC            STEP COUNTER
0025    770 ;
0025    771 ;           1           SC_30               1E
0025    772 ;           2           SC_14               0E
0025    773 ;           3           SC_6                06
0025    774 ;           4           SC_2                02
0025    775 ;
0025    776 ;
0025    777 ; LOGIC DESCRIPTION:
0025    778 ;
0025    779 ;       THIS TEST VERIFIES THAT THE PHB GATE ARRAY CAN SET THE STEP COUNTER
0025    780 ;       TO VALUES SPECIFIED BY THE MISC MICRO ORDER.  ERRORS SHOULD BE
0025    781 ;       CONFINED TO THE PHB ARRAY.
0025    782 ;
0025    783 ;
0025    784 ; ASSUMPTIONS:
0025    785 ;
0025    786 ;       THIS TEST ASSUMES THE STEP COUNTER HAS BEEN TESTED FROM THE WBUS.
0025    787 ;
0025    788 ;
0025    789 ; ERROR DESCRIPTION:
0025    790 ;
0025    791 ;       EXPECTED STEP COUNTER DATA
0025    792 ;       RECEIVED STEP COUNTER DATA
0025    793 ;       LOOP COUNT I
0025    794 ;
0025    795 ;--
0025    796 ;*****************************************************************************
0025    797
0025    798
0025    799         INITIALIZE                              ;INIT THE CPU
0027    800         LOOP            I,1,4                   ;CREATE A TEST LOOP OF 4
```

```
                                  002E          LOADDCS         1$,I,01,1                ;LOAD STEP COUNTER FUNCTION
                                  0035          ERRLOOP                                  ;ERROR LOOP POINT
                                  0037          FETCH           0                        ;START DCS PROGRAM
                                  003C          BRSTCLK         3                        ;END DCS PROGRAM
                                  0040          CMPREGMSK       WHREG,2$,I,3$,,L,        ;TEST STEP COUNTER CONTENTS
                                  004C          IFERROR         ,<PHB>,                  ;WRONG DATA
                                  0052          ENDLOOP         I                        ;END TEST LOOP
                                  0055          SKIP                                     ;GO TO NEXT TEST
                                  005C
                                  005C
                                  005C  BEGIN_TEST_DATA
                                  005C
                                  005C      ;-----------------------------------------------------------------------
                                  005C
                                  005C  1$:                                              ;MICRO CODE TABLE
U 01, 7700,4B80,0364,0300,0470,1802 005C  ;% 01   MISC/SC_30
U 01, 7700,CB00,0364,0300,0470,1802 0067  ;% 01   MISC/SC_14
U 01, 7700,CA80,0364,0300,0470,1802 0072  ;% 01   MISC/SC_6
U 01, 7700,4A00,0364,0300,0470,1802 007D  ;% 01   MISC/SC_2
                         0000001E 0088  2$:     .LONG   ^X0000001E                      ;RESULTS TABLE
                         0000000E 008C          .LONG   ^X0000000E
                         00000006 0090          .LONG   ^X00000006
                         00000002 0094          .LONG   ^X00000002
                         0000001F 0098  3$:     .LONG   ^X0000001F                      ;MASK FOR CMPREGMSK PSEUDO OP
                                  009C
                                  009C
                                  009C  BEGIN_CPU_DATA
                                  0002
                                  0002  DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801 0001  ;% 00   NOP
U 01, 7700,4B80,0364,0300,0470,1802 000C  ;% 01   MISC/SC_30                              ;SET-UP STEP COUNTER
U 02, 6700,C800,0364,0300,1870,1803 0017  ;% 02   RDM_WB,WCTRL/CM.TP.FPD.F5.STEPC         ;READ STEP COUNTER TO RDM
U 03, 7300,C800,0364,0300,0470,1804 0022  ;% 03   NOP,STOP.CLOCK                          ;STOP THE CPU CLOCK
                                  002D
                                  002D  END_CPU_DATA
                                  009C
                                  009C
                                  009C  END_TEST_DATA
                                  009C
                                  009C      ;-----------------------------------------------------------------------
                                  009C  ENDTEST
```

```
                        .SBTTL  "TEST   0BC     BRANCH ON STEP COUNTER TEST 1
                ;*******************************************************************************
                ;++
                ;
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       THIS TEST INSURES THAT THE BUT/DBZ.SC MICRO ORDER IS OPERATIONAL.
                ;
                ;
                ; TEST ALGORITHM:
                ;
                ;       1.  CREATE A TEST LOOP OF 2
                ;       2.  LOAD MICRO ORDER FOR THIS LOOP
                ;       3.  EXECUTE DCS PROGRAM
                ;               a.  SET STEP COUNTER TO 2
                ;               b.  BRANCH ON STEP COUNTER = 0
                ;       4.  TEST CS ADDRESS FOR CORRECT BRANCH
                ;       5.  IF NO ERROR GO TO STEP 2 AND NOP THE SC_2 MICRO INST.
                ;
                ;
                ; TEST PATTERNS:
                ;
                ;       NONE
                ;
                ;
                ; LOGIC DESCRIPTION:
                ;
                ;       THIS TEST INSURES THAT THE BUT/DBZ.SC MICRO ORDER CAN BE HANDLED
                ;       PROPERLY BY THE PHB GATE ARRAY.  ANY ERRORS SHOULD BE CONFINED TO
                ;       THE GATE ARRAY.
                ;
                ;
                ; ASSUMPTIONS:
                ;
                ;       THIS TEST ASSUMES THE STEP COUNTER CAN BE LOADED FROM THE MISC
                ;       MICRO ORDERS.
                ;
                ;
                ; ERROR DESCRIPTION:
                ;
                ;       EXPECTED CS ADDRESS
                ;       RECEIVED CS ADDRESS
                ;       LOOP COUNT I
                ;
                ;--
                ;*******************************************************************************
                
                
                        INITIALIZE                              ;INIT THE CPU
                        LOOP            I,1,2                   ;CREATE A TEST LOOP OF 2
                        LOADDCS         1$,I,01,1               ;LOAD MICRO CODE FOR THIS LOOP
                        ERRLOOP                                 ;ERROR LOOP POINT
                        FETCH           0                       ;START DCS PROGRAM
                        BRSTCLK         3                       ;END DCS PROGRAM
```

```
                                003D   918          CMPREGMSK       CSTRP,2$,I,3$,,W,       ;TEST CS ADDRESS
                                0049   919          IFERROR         ,<PHB>,                 ;ADDRESS INCORRECT
                                004F   920          ENDLOOP         I                       ;END TEST LOOP
                                0052   921          SKIP                                    ;GO TO NEXT TEST
                                0059   922
                                0059   923
                                0059   924 BEGIN_TEST_DATA
                                0059
                                0059          ;-------------------------------------------------------------------------
                                0059   925
                                0059   926 1$:                                              ;MICRO CODE FOR EACH LOOP
U 01, 7700,4A00,0364,0300,0470,1802   0059   927 ;% 01    MISC/SC_2
U 01, 7700,C800,0364,0300,0470,1802   0064   931 ;% 01    NOP
                              1800    006F   935 2$:      .WORD   ^X1800                  ;RESULTS TABLE
                              1801    0071   936          .WORD   ^X1801
                              3FFF    0073   937 3$:      .WORD   ^X3FFF                  ;MASK FOR CMPREGMSK PSEUDO OP
                                0075   938
                                0075   939
                                0075   940 BEGIN_CPU_DATA
                                0002   941
                                0002   942 DCS_DATA
                                0001   943
U 00, 7700,C800,0364,0300,0470,1801   0001   944 ;% 00    NOP
U 01, 7700,4A00,0364,0300,0470,1802   000C   948 ;% 01    MISC/SC_2                       ;SET STEP COUNTER TO 2
U 02, 7700,4800,0364,3300,0470,1800   0017   952 ;% 02    BUT/DBZ.SC,NEXT/1800            ;BRANCH ON STEP COUNTER = 0
U 03, 7300,C800,0364,0300,0470,1804   0022   956 ;% 03    NOP,STOP.CLOCK                  ;STOP THE CPU CLOCK
                                002D   960
                                002D   961 END_CPU_DATA
                                0075   962
                                0075   963
                                0075   964 END_TEST_DATA
                                0075
                                0075          ;-------------------------------------------------------------------------
                                0075   965 ENDTEST
```

```
                        .SBTTL  "TEST   0BD     BRANCH ON STEP COUNTER TEST 2
                    967 ;*****************************************************************************
                    968 ;++
                    969 ;
                    970 ; FUNCTIONAL DESCRIPTION:
                    971 ;
                    972 ;       THIS TEST CHEKCS THE BUT/BR.SC-4.INT-TS MICRO ORDER.
                    973 ;
                    974 ;
                    975 ; TEST ALGORITHM:
                    976 ;
                    977 ;       1.  CREATE A TEST LOOP OF 6
                    978 ;       2.  LOAD MICRO CODE FOR THIS LOOP
                    979 ;       3.  LOAD STEP COUNTER DATA INTO WDREG
                    980 ;       4.  EXECUTE DCS PROGRAM
                    981 ;               a.  SET A TIMER SERVICE INTERRUPT
                    982 ;               b.  LOAD TEST DATA INTO STEP COUNTER
                    983 ;               c.  BRANCH ON STEP COUNTER AND INTERRRUPT
                    984 ;       5.  TEST CS ADDRESS FOR CORRECT BRANCH
                    985 ;
                    986 ;
                    987 ; TEST PATTERNS:
                    988 ;
                    989 ;       ITERATION       TIMER SERVICE           STEP COUNTER    CS ADDRESS
                    990 ;
                    991 ;           1               SET                     05              1805
                    992 ;           2               CLEAR                   04              1804
                    993 ;           3               CLEAR                   03              1803
                    994 ;           4               CLEAR                   02              1803
                    995 ;           5               CLEAR                   01              1801
                    996 ;           6               CLEAR                   00              1800
                    997 ;
                    998 ;
                    999 ; LOGIC DESCRIPTION:
                   1000 ;
                   1001 ;       THIS TEST CHECKS THE VARIOUS BRANCHING CONDITIONS THAT THE
                   1002 ;       BUT/BR.SC-4.INT-TS MICRO ORDER CAN ENCOUNTER IN THE PHB GATE ARRAY.
                   1003 ;       ERRORS SHOULD BE CONFINED TO THE ARRAY.
                   1004 ;
                   1005 ;
                   1006 ; ASSUMPTIONS:
                   1007 ;
                   1008 ;       THIS TEST ASSUMES THE TIMER SERVICE INTERRUPT LOGIC IS OPERATIONAL.
                   1009 ;
                   1010 ;
                   1011 ; ERROR DESCRIPTION:
                   1012 ;
                   1013 ;       EXPECTED CS ADDRESS
                   1014 ;       RECEIVED CS ADDRESS
                   1015 ;       LOOP COUNT I
                   1016 ;
                   1017 ;--
                   1018 ;*****************************************************************************
                   1019
                   1020
```

```
                                0022  1021          INITIALIZE                                ;INIT THE CPU
                                0024  1022          LOOP            I,1,6                     ;CREATE A TEST LOOP OF 6
                                002B  1023          LOADDCS         1$,I,01,1                 ;LOAD MICRO CODE FOR THIS LOOP
                                0032  1024          LOADREG         WDREG,2$,I,L              ;LOAD DATA FOR STEP COUNTER
                                003A  1025          ERRLOOP                                   ;ERROR LOOP POINT
                                003C  1026          FETCH           0                         ;START DCS PROGRAM
                                0041  1027          BRSTCLK         4                         ;END DCS PROGRAM
                                0045  1028          CMPREGMSK       CSTRP,3$,I,4$,,W,         ;TEST CS ADDRESS
                                0051  1029          IFERROR         ,<PHB>,                   ;INCORRECT BRANCH
                                0057  1030          ENDLOOP         I                         ;END TEST LOOP I
                                005A  1031          SKIP                                      ;GO TO NEXT TEST
                                0061  1032
                                0061  1033
                                0061  1034 BEGIN_TEST_DATA
                                0061
                                0061      ;-------------------------------------------------------------------------
                                0061  1035
                                0061  1036 1$:                                               ;MICRO CODE TABLE
U 01, 7700,C800,5BE4,0300,1670,1802   0061  1037 ;% 01    WCTRL/TCSR_WB,WB_R[TEMP0]           ;SET TIMER SERVICE
U 01, 7700,C800,0364,0300,0410,1802   006C  1041 ;% 01    BUS/PRINIT                          ;CLEAR TIMER SERVICE
U 01, 7700,C800,0364,0300,0470,1802   0077  1045 ;% 01    NOP
U 01, 7700,C800,0364,0300,0470,1802   0082  1049 ;% 01    NOP
U 01, 7700,C800,0364,0300,0470,1802   008D  1053 ;% 01    NOP
U 01, 7700,C800,0364,0300,0470,1802   0098  1057 ;% 01    NOP
                          00000005    00A3  1061 2$:      .LONG   ^X00000005                  ;STEP COUNTER DATA
                          00000004    00A7  1062          .LONG   ^X00000004
                          00000003    00AB  1063          .LONG   ^X00000003
                          00000002    00AF  1064          .LONG   ^X00000002
                          00000001    00B3  1065          .LONG   ^X00000001
                          00000000    00B7  1066          .LONG   ^X00000000
                              1805    00BB  1067 3$:      .WORD   ^X1805                      ;RESULTS TABLE
                              1804    00BD  1068          .WORD   ^X1804
                              1803    00BF  1069          .WORD   ^X1803
                              1802    00C1  1070          .WORD   ^X1802
                              1801    00C3  1071          .WORD   ^X1801
                              1800    00C5  1072          .WORD   ^X1800
                              3FFF    00C7  1073 4$:      .WORD   ^X3FFF                      ;MASK FOR CMPREGMSK PSEUDO OP
                                      00C9  1074
                                      00C9  1075
                                      00C9  1076 BEGIN_CPU_DATA
                                      0002  1077
                                      0002  1078 DCS_DATA
                                      0001  1079
U 00, 7700,C800,0364,0300,0470,1801   0001  1080 ;% 00    NOP
U 01, 7700,C800,5BE4,0300,1670,1802   000C  1084 ;% 01    WCTRL/TCSR_WB,WB_R[TEMP0]           ;SET-UP TIMER SERVICE CONDITION
U 02, 5700,4800,0364,0300,1070,1803   0017  1088 ;% 02    WCTRL/STEPC_WB,WB_RDM               ;SET-UP STEP COUNTER CONDITION
U 03, 7700,0800,0364,3700,0470,1800   0022  1092 ;% 03    BUT/BR.SC-4.INT-TS,NEXT/1800        ;BRANCH ON STEP COUNTER
U 04, 7300,4800,0364,0300,0470,1805   002D  1096 ;% 04    NOP,STOP.CLOCK
                                      0038  1100
                                      0038  1101 RTEMP_DATA
                                      0001  1102
                          00100000    0001  1103          .LONG   ^X00100000                  ;SET TIMER SERVICE
                          00000000    0005  1104          .LONG   ^X00000000                  ;CLEAR TIMER SERVICE
                                      0009  1105
                                      0009  1106 END_CPU_DATA
```

```
                        00C9  1107
                        00C9  1108
                        00C9  1109 END_TEST_DATA
                        00C9
                        00C9       ;-----------------------------------------------------------------------
                        00C9  1110 ENDTEST
```

```
                        0027        .SBTTL  "TEST   0BE     TEST DECREMENTING THE STEP COUNTER
                        0027  1112 ;*******************************************************************************
                        0027  1113 ;++
                        0027  1114 ;
                        0027  1115 ; FUNCTIONAL DESCRIPTION:
                        0027  1116 ;
                        0027  1117 ;       THIS TEST CHECKS THE ABILITY OF THE PHB GATE ARRAY TO DECREMENT THE
                        0027  1118 ;       STEP COUNTER.
                        0027  1119 ;
                        0027  1120 ;
                        0027  1121 ; TEST ALGORITHM:
                        0027  1122 ;
                        0027  1123 ;       1.  CREATE A DECREMENTING LOOP OF 1F TO 1
                        0027  1124 ;       2.  SAVE THE LOOP COUNT IN 1$
                        0027  1125 ;       3.  MOVE THE LOOP COUNT TO THE WDREG
                        0027  1126 ;       4.  EXECUTE DCS PROGRAM
                        0027  1127 ;               a.  PUT THE LOOP COUNT IN RTEMP0
                        0027  1128 ;               b.  ADD 1 TO THE LOOP COUNT AND LOAD INTO THE STEP COUNTER
                        0027  1129 ;               c.  DECREMENT THE STEP COUNTER BY ONE
                        0027  1130 ;               d.  READ THE STEP COUNTER TO THE RDM
                        0027  1131 ;       5.  TEST THE STEP COUNTER VALUE
                        0027  1132 ;       6.  IF NO ERROR GO TO STEP 2 UNTIL THE LOOP IS EXHAUSTED
                        0027  1133 ;
                        0027  1134 ;
                        0027  1135 ; TEST PATTERNS:
                        0027  1136 ;
                        0027  1137 ;       NONE
                        0027  1138 ;
                        0027  1139 ;
                        0027  1140 ; LOGIC DESCRIPTION:
                        0027  1141 ;
                        0027  1142 ;       THIS TEST CHECKS THAT THE STEP COUNTER CAN BE DECREMENTED FROM
                        0027  1143 ;       IT'S FULL VALUE DOWN TO ZERO.  ANY ERRORS SHOULD BE IN THE PHB
                        0027  1144 ;       GATE ARRAY.
                        0027  1145 ;
                        0027  1146 ;
                        0027  1147 ; ASSUMPTIONS:
                        0027  1148 ;
                        0027  1149 ;       THIS TEST ASSUMES THE ALU CAN DO ADDS.
                        0027  1150 ;
                        0027  1151 ;
                        0027  1152 ; ERROR DESCRIPTION:
                        0027  1153 ;
                        0027  1154 ;       EXPECTED STEP COUNTER DATA
                        0027  1155 ;       RECEIVED STEP COUNTER DATA
                        0027  1156 ;       LOOP COUNT I
                        0027  1157 ;
                        0027  1158 ;--
                        0027  1159 ;*******************************************************************************
                        0027  1160
                        0027  1161
                        0027  1162         INITIALIZE                              ;INIT THE CPU
                        0029  1163         LOOP            I,<^X1F>,1              ;CREATE A DECREMENTING LOOP
                        0030  1164         SAVEINDEX       I,1$                    ;STORE LOOP COUNT
                        0035  1165         LOADREG         WDREG,1$,,L             ;PUT LOOP COUNT IN WDREG
```

```
                                  003D          ERRLOOP                                  ;ERROR LOOP POINT
                                  003F          FETCH          0                         ;START DCS PROGRAM
                                  0044          BRSTCLK        5                         ;END DCS PROGRAM
                                  0048          CMPREGMSK      WHREG,1$,,2$,,L,          ;TEST STEP COUNTER CONTENTS
                                  0054          IFERROR        ,<PHB>,                   ;INCORRECT DATA
                                  005A          ENDLOOP        I                         ;END TEST LOOP
                                  005D          SKIP                                     ;GO TO NEXT TEST
                                  0064
                                  0064
                                  0064  BEGIN_TEST_DATA
                                  0064
                                  0064      ;-----------------------------------------------------------------------
                                  0064
                        00000000  0064  1$:     .LONG   ^X00000000                       ;LOOP COUNT STORAGE
                        0000001F  0068  2$:     .LONG   ^X0000001F                       ;MASK FOR CMPREGMSK PSEUDO OP
                                  006C
                                  006C
                                  006C
                                  006C  BEGIN_CPU_DATA
                                  0002
                                  0002  DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 5700,8840,0364,0300,0470,1802  000C  ;% 01   R[TEMP0]_RDM                            ;MOVE LOOP COUNT TO RTEMP0
U 02, 7700,0800,E7C0,0300,1070,1803  0017  ;% 02   WCTRL/STEPC_WB,WB_R[TEMP0]+1             ;LOOP + 1 TO STEP COUNTER
U 03, 7700,C980,0364,0300,0470,1804  0022  ;% 03   MISC/DEC.SC                             ;DECREMENT STEP COUNTER
U 04, 6700,C800,0364,0300,1870,1805  002D  ;% 04   RDM_WB,WCTRL/CM.TP.FPD.F5.STEPC          ;READ STEP COUNTER TO REM
U 05, 7300,4800,0364,0300,0470,1806  0038  ;% 05   NOP,STOP.CLOCK                          ;STOP THE CPU CLOCK
                                  0043
                                  0043  END_CPU_DATA
                                  006C
                                  006C
                                  006C  END_TEST_DATA
                                  006C
                                  006C      ;-----------------------------------------------------------------------
                                  006C  ENDTEST
```

```
                        0017        .SBTTL  "TEST   0BF     TEST BRANCH ON FPD
                        0017  1217 ;*******************************************************************************
                        0017  1218 ;++
                        0017  1219 ;
                        0017  1220 ; FUNCTIONAL DESCRIPTION:
                        0017  1221 ;
                        0017  1222 ;       TEST BRANCHING ON PSL <27> FPD BIT
                        0017  1223 ;
                        0017  1224 ;
                        0017  1225 ; TEST ALGORITHM:
                        0017  1226 ;
                        0017  1227 ;       1.  CREATE A TEST LOOP OF 2
                        0017  1228 ;       2.  LOAD MISC MICRO ORDER FOR THIS LOOP
                        0017  1229 ;       3.  EXECUTE DCS PROGRAM
                        0017  1230 ;               a.  CONDITION FPD BIT
                        0017  1231 ;               b.  BRANCH ON FPD BIT
                        0017  1232 ;       4.  TEST CS ADDRESS FOR CORRECT BRANCH
                        0017  1233 ;       5.  IF NO ERROR GO TO STEP 2 UNTIL LOOP EXHAUSTED
                        0017  1234 ;
                        0017  1235 ;
                        0017  1236 ; TEST PATTERNS:
                        0017  1237 ;
                        0017  1238 ;       NONE
                        0017  1239 ;
                        0017  1240 ;
                        0017  1241 ; LOGIC DESCRIPTION:
                        0017  1242 ;
                        0017  1243 ;       THIS TEST CHECKS THE ABILITY OF THE PHB GATE ARRAY TO BRANCH THE
                        0017  1244 ;       CS ADDRESS BASED ON THE CONDITION OF THE FPD BIT.  ALL ERRORS SHOULD
                        0017  1245 ;       BE CONFINED TO THE ARRAY.
                        0017  1246 ;
                        0017  1247 ;
                        0017  1248 ; ASSUMPTIONS:
                        0017  1249 ;
                        0017  1250 ;       THIS TEST ASSUMES THE PSL HAS BEEN TESTED.
                        0017  1251 ;
                        0017  1252 ;
                        0017  1253 ; ERROR DESCRIPTION:
                        0017  1254 ;
                        0017  1255 ;       EXPECTED CS ADDRESS
                        0017  1256 ;       RECEIVED CS ADDRESS
                        0017  1257 ;       LOOP COUNT I
                        0017  1258 ;
                        0017  1259 ;--
                        0017  1260 ;*******************************************************************************
                        0017  1261
                        0017  1262
                        0017  1263          INITIALIZE                              ;INIT THE CPU
                        0019  1264          LOOP            I,1,2                   ;CREATE A TEST LOOP OF 2
                        0020  1265          LOADDCS         1$,I,01,1               ;LOAD MICRO CODE FOR THIS LOOP
                        0027  1266          ERRLOOP                                 ;ERROR LOOP POINT
                        0029  1267          FETCH           0                       ;START DCS PROGRAM
                        002E  1268          BRSTCLK         3                       ;END DCS PROGRAM
                        0032  1269          CMPREGMSK       CSTRP,2$,I,3$,,W,       ;TEST CS ADDRESS
                        003E  1270          IFERROR         ,<PHB>,                 ;INCORRECT BRANCH ADDRESS
```

```
                                  0044          ENDLOOP         I                        ;END TEST LOOP I
                                  0047          SKIP                                     ;GO TO NEXT TEST
                                  004E
                                  004E
                                  004E  BEGIN_TEST_DATA
                                  004E
                                  004E        ;-----------------------------------------------------------------------
                                  004E
                                  004E  1$:                                              ;MICRO CODE TABLE
U 01, 7700,4E80,0364,0300,0470,1802 004E  ;% 01   MISC/SET.FPD
U 01, 7700,CE00,0364,0300,0470,1802 0059  ;% 01   MISC/CLR.FPD
                             1801 0064  2$:     .WORD   ^X1801                          ;RESULTS TABLE
                             1800 0066          .WORD   ^X1800
                             3FFF 0068  3$:     .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                                  006A
                                  006A
                                  006A  BEGIN_CPU_DATA
                                  0002
                                  0002  DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801 0001  ;% 00   NOP
U 01, 7700,4E80,0364,0300,0470,1802 000C  ;% 01   MISC/SET.FPD                            ;SET-UP FPD BIT
U 02, 7700,4800,0364,3F00,0470,1800 0017  ;% 02   BUT/FPD,NEXT/1800                       ;BRANCH ON FPD BIT
U 03, 7300,C800,0364,0300,0470,1804 0022  ;% 03   NOP,STOP.CLOCK                          ;STOP THE CPU CLOCK
                                  002D
                                  002D  END_CPU_DATA
                                  006A
                                  006A
                                  006A  END_TEST_DATA
                                  006A
                                  006A        ;-----------------------------------------------------------------------
                                  006A  ENDTEST
```

```
                        001B        .SBTTL  "TEST   0C0     TEST BRANCH ON MMNOINT
                        001B  1318 ;*******************************************************************************
                        001B  1319 ;++
                        001B  1320 ;
                        001B  1321 ; FUNCTIONAL DESCRIPTION:
                        001B  1322 ;
                        001B  1323 ;       THIS TEST CHECKS THE OPERATION OF THE BUT/MM.ALLOW.INT MICRO ORDER.
                        001B  1324 ;
                        001B  1325 ;
                        001B  1326 ; TEST ALGORITHM:
                        001B  1327 ;
                        001B  1328 ;       1.  CREATE A TEST LOOP OF 3
                        001B  1329 ;       2.  LOAD MICRO CODE FOR THIS LOOP
                        001B  1330 ;       3.  EXECUTE DCS PROGRAM
                        001B  1331 ;               a.  SET UP TIMER SERVICE CONDITION
                        001B  1332 ;               b.  SET UP FLAG4 CONDITION
                        001B  1333 ;               c.  BRANCH ON FLAG4 AND INTERRUPT
                        001B  1334 ;       4.  TEST CS ADDRESS FOR CORRECT BRANCH
                        001B  1335 ;       5.  IF NO ERROR GO TO STEP 2 UNTIL LOOP IS EXHAUSTED
                        001B  1336 ;
                        001B  1337 ;
                        001B  1338 ; TEST PATTERNS:
                        001B  1339 ;
                        001B  1340 ;       NONE
                        001B  1341 ;
                        001B  1342 ;
                        001B  1343 ; LOGIC DESCRIPTION:
                        001B  1344 ;
                        001B  1345 ;       THIS TEST CHECKS THE VARIOUS CONDITIONS OF FLAG4 AND INTERRUPT TO
                        001B  1346 ;       DETERMINE IF THE PHB GATE ARRAY CAN BRANCH PROPERLY.  ALL ERRORS
                        001B  1347 ;       SHOULD BE IN THE PHB ARRAY.
                        001B  1348 ;
                        001B  1349 ;
                        001B  1350 ; ASSUMPTIONS:
                        001B  1351 ;
                        001B  1352 ;       THIS TEST ASSUMES THE TIMER SERVICE INTERRUPT LOGIC IS OPERATIONAL.
                        001B  1353 ;
                        001B  1354 ;
                        001B  1355 ; ERROR DESCRIPTION:
                        001B  1356 ;
                        001B  1357 ;       EXPECTED CS ADDRESS
                        001B  1358 ;       RECEIVED CS ADDRESS
                        001B  1359 ;       LOOP COUNT I
                        001B  1360 ;
                        001B  1361 ;--
                        001B  1362 ;*******************************************************************************
                        001B  1363
                        001B  1364
                        001B  1365         INITIALIZE                              ;INIT THE CPU
                        001D  1366         LOOP            I,1,3                   ;CREATE A TEST LOOP OF 3
                        0024  1367         LOADDCS         1$,I,01,2               ;LOAD MICRO CODE FOR THIS LOOP
                        002B  1368         ERRLOOP                                 ;ERROR LOOP POINT
                        002D  1369         FETCH           0                       ;START DCS PROGRAM
                        0032  1370         BRSTCLK         4                       ;END DCS PROGRAM
                        0036  1371         CMPREGMSK       CSTRP,2$,I,3$,,W,       ;TEST CS ADDRESS
```

```
                                  0042          IFERROR         ,<PHB>,                  ;INCORRECT BRANCH
                                  0048          ENDLOOP         I                        ;END TEST LOOP I
                                  004B          SKIP                                     ;GO TO NEXT TEST
                                  0052
                                  0052
                                  0052  BEGIN_TEST_DATA
                                  0052
                                  0052      ;-----------------------------------------------------------------------
                                  0052
                                  0052  1$:                                              ;MICRO CODE TABLE
U 01, 7700,C800,5BE4,0300,1670,1802 0052  ;% 01   WCTRL/TCSR_WB,WB_R[TEMP0]              ;SET TIMER SERVICE
U 02, 7700,4200,0364,0300,0470,1803 005D  ;% 02   MISC/CLR.MMNOINT                       ;CLEAR FLAG4
U 01, 7700,C800,0364,0300,0410,1802 0068  ;% 01   BUS/PRINIT                             ;CLEAR TIMER SERVICE
U 02, 7700,4200,0364,0300,0470,1803 0073  ;% 02   MISC/CLR.MMNOINT                       ;CLEAR FLAG4
U 01, 7700,C800,5BE4,0300,1670,1802 007E  ;% 01   WCTRL/TCSR_WB,WB_R[TEMP0]              ;SET TIMER SERVICE
U 02, 7700,C600,0364,0300,0470,1803 0089  ;% 02   MISC/SET.MMNOINT                       ;SET FLAG4
                              1802  0094  2$:     .WORD   ^X1802                         ;RESULTS TABLE
                              1800  0096          .WORD   ^X1800
                              1800  0098          .WORD   ^X1800
                              3FFF  009A  3$:     .WORD   ^X3FFF                         ;MASK FOR CMPREGMSK PSEUDO OP
                                  009C
                                  009C
                                  009C  BEGIN_CPU_DATA
                                  0002
                                  0002  DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801 0001  ;% 00   NOP
U 01, 7700,C800,5BE4,0300,1670,1802 000C  ;% 01   WCTRL/TCSR_WB,WB_R[TEMP0]              ;SET-UP TIMER SERVICE CONDITION
U 02, 7700,4200,0364,0300,0470,1803 0017  ;% 02   MISC/CLR.MMNOINT                       ;SET-UP FLAG4 CONDITION
U 03, 7700,0800,0364,2F00,0470,1800 0022  ;% 03   BUT/MM.ALLOW.INT,NEXT/1800             ;BRANCH ON CONDITIONS
U 04, 7300,4800,0364,0300,0470,1805 002D  ;% 04   NOP,STOP.CLOCK                         ;STOP THE CPU CLOCK
                                  0038
                                  0038  RTEMP_DATA
                                  0001
                          00100000  0001          .LONG   ^X00100000                     ;SET TIMER SERVICE
                                  0005
                                  0005  END_CPU_DATA
                                  009C
                                  009C
                                  009C  END_TEST_DATA
                                  009C
                                  009C      ;-----------------------------------------------------------------------
                                  009C  ENDTEST
```

```
                        0028        .SBTTL  "TEST   0C1     CLEAR STEP COUNTER AND FLAGS ON IRD
                        0028  1445 ;*******************************************************************************
                        0028  1446 ;++
                        0028  1447 ;
                        0028  1448 ; FUNCTIONAL DESCRIPTION:
                        0028  1449 ;
                        0028  1450 ;       THIS TEST VERIFIES THAT THE STEP COUNTER AND STATUS FLAGS ARE CLEARED
                        0028  1451 ;       ON IRD.
                        0028  1452 ;
                        0028  1453 ;
                        0028  1454 ; TEST ALGORITHM:
                        0028  1455 ;
                        0028  1456 ;       1.  CREATE A TEST LOOP OF 2
                        0028  1457 ;       2.  LOAD MICRO CODE FOR THIS LOOP
                        0028  1458 ;       3.  EXECUTE DCS PROGRAM
                        0028  1459 ;               a.  LOAD PC WITH ZERO TO CAUSE PREFETCH
                        0028  1460 ;               b.  DISABLE CONTROL P INTERRUPTS
                        0028  1461 ;               c.  WRITE ONES TO THE STEP COUNTER AND STATUS FLAGS
                        0028  1462 ;               d.  PERFORM AN IRD OPERATION
                        0028  1463 ;               e.  READ THE STEP COUNTER OR STATUS FLAGS TO THE RDM
                        0028  1464 ;       4.  TEST DATA FOR ZEROS
                        0028  1465 ;       5.  IF NO ERROR GO TO STEP 2 UNTIL LOOP IS EXHAUSTED
                        0028  1466 ;
                        0028  1467 ;
                        0028  1468 ; TEST PATTERNS:
                        0028  1469 ;
                        0028  1470 ;       NONE
                        0028  1471 ;
                        0028  1472 ;
                        0028  1473 ; LOGIC DESCRIPTION:
                        0028  1474 ;
                        0028  1475 ;       THIS TEST INSURES THAT THE STEP COUNTER AND STATUS FLAGS IN THE PHB
                        0028  1476 ;       GATE ARRAY ARE ZEROED BY AN IRD OPERATION.  ALL ERRORS SHOULD BE
                        0028  1477 ;       IN THE PHB ARRAY.
                        0028  1478 ;
                        0028  1479 ;
                        0028  1480 ; ASSUMPTIONS:
                        0028  1481 ;
                        0028  1482 ;       THIS TEST ASSUMES THE STEP COUNTER AND STATUS FLAGS HAVE BEEN TESTED
                        0028  1483 ;       VIA THE WBUS.
                        0028  1484 ;
                        0028  1485 ;
                        0028  1486 ; ERROR DESCRIPTION:
                        0028  1487 ;
                        0028  1488 ;       EXPECTED REGISTER CONTENTS
                        0028  1489 ;       RECEIVED REGISTER CONTENTS
                        0028  1490 ;       LOOP COUNT I
                        0028  1491 ;
                        0028  1492 ;--
                        0028  1493 ;*******************************************************************************
                        0028  1494
                        0028  1495
                        0028  1496         INITIALIZE                              ;INIT THE CPU
                        002A  1497         LOOP            I,1,2                   ;CREATE A TEST LOOP OF 2
                        0031  1498         LOADDCS         1$,I,07,1               ;LOAD MICRO CODE FOR THIS LOOP
```

```
                                  0038          ERRLOOP                                  ;ERROR LOOP POINT
                                  003A          FETCH          0                         ;START DCS PROGRAM
                                  003F          BRSTCLK        8                         ;END DCS PROGRAM
                                  0043          CMPREGMSK      WHREG,2$,,3$,I,L          ;TEST FOR CLEAR REGISTERS
                                  004F          IFERROR        ,<PHB>,                   ;REGISTER DID NOT INIT
                                  0055          ENDLOOP        I                         ;END TEST LOOP I
                                  0058          SKIP                                     ;GO TO NEXT TEST
                                  005F
                                  005F
                                  005F  BEGIN_TEST_DATA
                                  005F
                                  005F      ;-----------------------------------------------------------------------
                                  005F
                                  005F  1$:                                              ;MICRO CODE FOR EACH LOOP
U 07, 6700,4800,0364,0300,1870,1808  005F  ;% 07   RDM_WB,WCTRL/CM.TP.FPD.F5.STEPC          ;READ STEP COUNTER
U 07, 6700,C800,0364,0300,3870,1808  006A  ;% 07   RDM_WB,WCTRL/CM.TP.FPD.FLAGS             ;READ FLAGS
                         00000000  0075  2$:     .LONG   ^X00000000                       ;EXPECTED RESULT
                         0000001F  0079  3$:     .LONG   ^X0000001F                       ;MASK FOR CMPREGMSK PSEUDO OP
                         0000003F  007D          .LONG   ^X0000003F                       ;...
                                  0081
                                  0081
                                  0081  BEGIN_CPU_DATA
                                  0002
                                  0002  DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 7700,4800,5BE4,03D8,4870,1802  000C  ;% 01   PC_R[ZERO]                               ;CAUSE PREFETCH
U 02, 7700,C800,5BE4,0300,2470,1803  0017  ;% 02   WCTRL/LOADCRAR,WB_R[TEMP0]               ;DISABLE ^P INTERRUPTS
U 03, 7700,8800,5BE4,0304,2C70,1804  0022  ;% 03   WCTRL/CONWRITE,WB_R[TEMP1]               ;...
U 04, 7700,0800,5BE4,0308,1070,1805  002D  ;% 04   WCTRL/STEPC_WB,WB_R[TEMP2]               ;WRITE ONES TO STEP COUNTER
U 05, 7700,8800,5BE4,0308,3070,1806  0038  ;% 05   WCTRL/FLAGS_WB,WB_R[TEMP2]               ;WRITE ONES TO THE FLAGS
U 06, 7700,C800,0364,1700,0470,1807  0043  ;% 06   IRD1TEST                                 ;PERFORM IRD
U 07, 6700,4800,0364,0300,1870,1808  004E  ;% 07   RDM_WB,WCTRL/CM.TP.FPD.F5.STEPC          ;READ REGISTER TO RDM
U 08, 7300,4800,0364,0300,0470,1809  0059  ;% 08   NOP,STOP.CLOCK                           ;STOP THE CPU CLOCK
                                  0064
                                  0064  RTEMP_DATA
                                  0001
                         00C00000  0001          .LONG   ^X00C00000                       ;DISABLE ^P INTERRUPTS
                         00400000  0005          .LONG   ^X00400000                       ;...
                         0000003F  0009          .LONG   ^X0000003F                       ;WRITE ONE'S TO REGISTERS
                                  000D
                                  000D  CACHE_DATA
                                  0001
                         00000000  0001          .LONG   ^X00000000                       ;VALIDATE LOCATIONS
                         00000000  0005          .LONG   ^X00000000                       ;...
                                  0009
                                  0009  END_CPU_DATA
                                  0081
                                  0081
                                  0081  END_TEST_DATA
                                  0081
                                  0081      ;-----------------------------------------------------------------------
                                  0081
                                  0081  ENDTEST
```

```
                        002B        .SBTTL  "TEST   0C2     IRD1 Instruction Address Decode Test 1
                        002B  1583 ;*****************************************************************************
                        002B  1584 ;++
                        002B  1585 ;
                        002B  1586 ; FUNCTIONAL DESCRIPTION
                        002B  1587 ;
                        002B  1588 ;       This test will examine the ROMS that implement a native mode IRD1 when
                        002B  1589 ;       the First Part Done (FPD) bit of the PSL is set to zero.
                        002B  1590 ;
                        002B  1591 ; TEST ALGORITHM
                        002B  1592 ;
                        002B  1593 ;       This test is implemented in three subtests. The first subtest examines
                        002B  1594 ;       the IRD1 ROMS when the CPU's Floating Point Accelerator (FPA) is not
                        002B  1595 ;       present. The second subtest examines the ROMS when there is an FPA
                        002B  1596 ;       present and it has been disabled. The third subtest examines the ROMS
                        002B  1597 ;       when the FPA is present and enabled. The test algorithm executes as
                        002B  1598 ;       follows:
                        002B  1599 ;
                        002B  1600 ;        1. Initialize the CPU. This will clear the FPD bit in the PSL.
                        002B  1601 ;        2. Execute the following microcode to initialize an FPA:
                        002B  1602 ;                1. NOP to get the CPU up to speed.
                        002B  1603 ;                2. Load four NOP opcodes into the longlit register for XB0.
                        002B  1604 ;                3. Zero the VA.
                        002B  1605 ;                4. Write the NOPS for XB0 into cache at address 0.
                        002B  1606 ;                5. Increment the VA by 4.
                        002B  1607 ;                6. Validate the next location for XB1 prefetch by writing
                        002B  1608 ;                   NOPS to cache at address 4.
                        002B  1609 ;                7. Zero the PC to initiate the instruction prefetch that
                        002B  1610 ;                   writes XB0 and XB1 from cache.
                        002B  1611 ;                8. Execute an IRD1TEST cycle to decode the NOP in XB0.
                        002B  1612 ;                   This will prepare an FPA to be enabled or disabled.
                        002B  1613 ;                9. Halt.
                        002B  1614 ;        3. Execute the following microcode to test for an FPA presence:
                        002B  1615 ;                10. NOP to get the CPU up to speed.
                        002B  1616 ;                11. BUT/NO.FPA,NEXT/1800
                        002B  1617 ;                12. STOP.CLOCK to halt.
                        002B  1618 ;        4. Perform a masked comparison of the address in the Control Store
                        002B  1619 ;           Address Trace Register (CSTRP) against the expected address for the
                        002B  1620 ;           FPA not present.
                        002B  1621 ;        5. If the two do not match, then there is an FPA so branch to
                        002B  1622 ;           Subtest 2. Otherwise continue on to Subtest 1.
                        002B  1623 ;
                        002B  1624 ;       Subtest 1
                        002B  1625 ;
                        002B  1626 ;        6. Loop on error from here.
                        002B  1627 ;        7. Re-execute the microcode in step 3 to test the FPA enable bit.
                        002B  1628 ;        8. Perform a masked comparison of the CSTRP with the expected address
                        002B  1629 ;           for the FPA disabled.
                        002B  1630 ;        9. If the two do not match then there is a problem with the FPA/CPU
                        002B  1631 ;           interface so signal an error.
                        002B  1632 ;       10. Prepare the RDM to expect non-DCS addresses.
                        002B  1633 ;       11. Prepare a loop on I for 256 VAX opcodes.
                        002B  1634 ;       12. Save the loop counter minus one in the RDM memory.
                        002B  1635 ;       13. Load the WDREG with the number saved in RDM memory.
                        002B  1636 ;       14. Loop on error from here.
```

```
002B  1637 ;       15. Execute the following test microcode:
002B  1638 ;             13. NOP to get the CPU up to speed.
002B  1639 ;             14. Load the opcode to be tested from the WDREG to RTEMP1.
002B  1640 ;             15. Load the operands into the longlit register.
002B  1641 ;             16. Move the operands from the longlit register into the
002B  1642 ;                 Q register.
002B  1643 ;             17. Form the test instruction by ORing the opcode in RTEMP1
002B  1644 ;                 (MTEMP1=RTEMP1) with the operands in the Q register and
002B  1645 ;                 write the result back to RTEMP1.
002B  1646 ;             18. Zero the VA.
002B  1647 ;             19. Write the test instruction in RTEMP1 to cache at address 0.
002B  1648 ;             20. Increment the VA by 4.
002B  1649 ;             21. Validate the next location for XB1 prefetch by writing
002B  1650 ;                 zeros to cache at address 4.
002B  1651 ;             22. Zero the PC to initiate the instruction prefetch that
002B  1652 ;                 writes XB0 and XB1 from cache.
002B  1653 ;             23. Write the contents of XB0 to RTEMP0. This will cause the
002B  1654 ;                 PC to increment by 4 and can change the contents of XB0.
002B  1655 ;             24. Rezero the PC to reload XB0 and XB1.
002B  1656 ;             25. Execute a BUT/IRD1 to decode the instruction in XB0.
002B  1657 ;             26. Halt.
002B  1658 ;       16. Perform a masked comparison of the address in the CSTRP register
002B  1659 ;           with the expected control store address for the IRD1.
002B  1660 ;       17. If the two are not equal, signal an error.
002B  1661 ;       18. Continue to loop on I for the remaining VAX opcodes.
002B  1662 ;       19. Skip to the next test.
002B  1663 ;
002B  1664 ;       Subtests 2 and 3
002B  1665 ;
002B  1666 ;       20. Load the WDREG with a longword to perform an FPA disable for
002B  1667 ;           Subtest 2 or an FPA enable for Subtest 3.
002B  1668 ;       21. Loop on error from here.
002B  1669 ;       22. Execute the following microcode to disable the FPA for Subtest 2 or
002B  1670 ;           to enable the FPA for Subtest 3 and to also check the result:
002B  1671 ;                26. NOP to get the CPU up to speed.
002B  1672 ;                27. WCTRL/FPAENABLE_WB5,WB_RDM to write the FPA's enable bit
002B  1673 ;                    with bit 5 of the RDM's WDREG.
002B  1674 ;                28. NOP
002B  1675 ;                29. BUT/NO.FPA,NEXT/1800 to test for FPA enabled.
002B  1676 ;                30. STOP.CLOCK to halt.
002B  1677 ;       23. Perform a masked comparison of the CSTRP with the expected address
002B  1678 ;           for the FPA disabled (Subtest 2) or the FPA enabled (Subtest 3).
002B  1679 ;       24. If the two do not match then there is a problem with the FPA/CPU
002B  1680 ;           interface so signal an error.
002B  1681 ;       25. Prepare the RDM to expect non-DCS addresses.
002B  1682 ;       26. Prepare a loop on I for 256 VAX opcodes.
002B  1683 ;       27. Save the loop counter minus one in the RDM memory.
002B  1684 ;       28. Load the WDREG with the number saved in RDM memory.
002B  1685 ;       29. Loop on error from here.
002B  1686 ;       30. Execute the same test microcode as in Subtest 1 step 15.
002B  1687 ;       31. Perform a masked comparison of the address in the CSTRP register
002B  1688 ;           with the expected control store address for the IRD1.
002B  1689 ;       32. If the two are not equal, signal an error.
002B  1690 ;       33. Continue to loop on I for the remaining VAX opcodes.
002B  1691 ;       34. If this is Subtest 2 then continue on to Subtest 3.
```

```
                     1692 ;          If this is Subtest 3 then skip to the next test.
                     1693 ;
                     1694 ; TEST PATTERNS
                     1695 ;
                     1696 ;       For this test the VAX machine opcodes are generated by a simple count
                     1697 ;       pattern. The opcodes are based on the 256 base machine instructions
                     1698 ;       as specified in the VAX architecture handbook. The generated opcodes
                     1699 ;       are written to the CPU and combined with three bytes of operand
                     1700 ;       data to form a VAX machine instruction. The operands all specify R0 in
                     1701 ;       register mode and are loaded from the CPU's long literal register.
                     1702 ;
                     1703 ;       Also required are two tables of expected control store addresses for
                     1704 ;       each IRD1 Instruction Decode Cycle. One table contains the expected
                     1705 ;       addresses for an IRD1 when the FPA has been disabled, the other for
                     1706 ;       when it is enabled.  The expected control store addresses were obtained
                     1707 ;       from the Native Mode Ird Rom Definition in the CPU microcode.
                     1708 ;
                     1709 ; LOGIC DESCRIPTION
                     1710 ;
                     1711 ;       The CCS ROMS are fed addresses from ROMS on the DPM during instruction
                     1712 ;       decode. The DPM ROMS are addressed by bits <07:00> of the execution
                     1713 ;       buffer. These bits form the opcode for the instruction. For zero and
                     1714 ;       certain single operand instructions the addresses generated by the ROMS
                     1715 ;       will force a jump to execution flows in the microcode. For the rest,
                     1716 ;       operand specifier evaulation addresses will be generated.
                     1717 ;
                     1718 ; ASSUMPTIONS
                     1719 ;
                     1720 ;       This test assumes the following logic has already been tested; the
                     1721 ;       CPU's data path, the cache, the XB, and the IR and OSR.
                     1722 ;
                     1723 ; ERROR DESCRIPTION
                     1724 ;
                     1725 ;       On error this test can type out the following information:
                     1726 ;
                     1727 ;       Failing gate arrays:    IRD
                     1728 ;       Failing modules:        FPA, MIC
                     1729 ;
                     1730 ;       The following data will also by typed:
                     1731 ;
                     1732 ;       EXPECTED CONTROL STORE ADDRESS
                     1733 ;       RECEIVED CONTROL STORE ADDRESS
                     1734 ;       LOOP COUNT (INDEX I)
                     1735 ;       THE INSTRUCTION BEING DECODED FROM XB0 (RTEMP0)
                     1736 ;
                     1737 ;--
                     1738 ;*******************************************************************************
                     1739
002B                 1740         INITIALIZE                                 ; INIT THE CPU AND CLEAR FPD
002D                 1741         FETCH           <^X15>                     ; PREPARE AN FPA TO BE ENABLED
0032                 1742         BRSTCLK         <^X1F>                     ; OR DISABLED
0036                 1743         FETCH           <^X12>                     ; EXECUTE A BUT/NO.FPA TO TEST
003B                 1744         BRSTCLK         <^X14>                     ; FOR AN FPA PRESENT
003F                 1745         CMPREGMSK       CSTRP,1$,,2$,,W            ; CHECK THE BUT/NO.FPA ADDRESS
004B                 1746         SKIP            100$,ONERROR               ; SKIP IF THE FPA IS PRESENT
```

```
                        0052  1747
                        0052  1748          SUBTEST         1
                        0055
                        0055        ;////////////////////////////////////////////////////////////////////////////
                        0055  1749  ;+
                        0055  1750  ; Test the IRD1 ROMS when the FPA is not present.
                        0055  1751  ;-
                        0055  1752          ERRLOOP                                 ; LOOP ON ERROR HERE
                        0057  1753          FETCH           <^X12>                  ; EXECUTE A BUT/NO.FPA TO TEST
                        005C  1754          BRSTCLK         <^X14>,INH              ; FOR THE FPA DISABLED
                        0060  1755          CMPREGMSK       CSTRP,3$,,5$,,W         ; IS THE FPA DISABLED?
                        006C  1756          IFERROR         ,,<FPA>                 ; IF NOT, SIGNAL AN ERROR
                        0072  1757          EXPUTRAP                                ; ELSE EXPECT NON-DCS ADDRESSES
                        0074  1758          LOOP            I,1,256                 ; LOOP FOR 256 OPCODES
                        007B  1759          SAVEINDEX       I,9$                    ; SAVE I-1 TO FORM AN OPCODE
                        0080  1760          LOADREG         WDREG,9$,,L             ; LOAD WDREG WITH THE OPCODE
                        0088  1761          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        008A  1762          FETCH           0                       ; EXECUTE THE IRD1 ADDRESS TEST
                        008F  1763          BRSTCLK         <^X0F>,INH              ; ...
                        0093  1764          CMPREGMSK       CSTRP,10$,I,6$,,W       ; IS THE CS ADDRESS CORRECT?
                        009F  1765          IFERROR         1,<IRD>,<MIC>           ; IF NOT, SIGNAL AN ERROR
                        00A6  1766          ENDLOOP         I                       ; CONTINUE THE LOOP
                        00A9  1767          SKIP                                    ; SKIP TO THE NEXT TEST
                        00B0  1768
                        00B0  1769  100$:   SUBTEST         2
                        00B3
                        00B3        ;////////////////////////////////////////////////////////////////////////////
                        00B3  1770  ;+
                        00B3  1771  ; Test the IRD1 ROMS with the FPA present but not enabled.
                        00B3  1772  ;-
                        00B3  1773          LOADREG         WDREG,7$,,L             ; LOAD DATA TO DISABLE THE FPA
                        00BB  1774          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        00BD  1775          FETCH           <^X10>                  ; EXECUTE AN FPA DISABLE AND
                        00C2  1776          BRSTCLK         <^X14>,INH              ; READ BACK THE FPA STATUS
                        00C6  1777          CMPREGMSK       CSTRP,3$,,5$,,W         ; IS THE FPA DISABLED?
                        00D2  1778          IFERROR         ,,<FPA>                 ; IF NOT, SIGNAL AN ERROR
                        00D8  1779          EXPUTRAP                                ; EXPECT NON-DCS ADDRESSES
                        00DA  1780          LOOP            I,1,256                 ; LOOP FOR 256 OPCODES
                        00E1  1781          SAVEINDEX       I,9$                    ; SAVE I-1 TO FORM AN OPCODE
                        00E6  1782          LOADREG         WDREG,9$,,L             ; LOAD WDREG WITH THE OPCODE
                        00EE  1783          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        00F0  1784          FETCH           0                       ; EXECUTE THE IRD1 ADDRESS TEST
                        00F5  1785          BRSTCLK         <^X0F>,INH              ; ...
                        00F9  1786          CMPREGMSK       CSTRP,10$,I,6$,,W       ; IS THE CS ADDRESS CORRECT?
                        0105  1787          IFERROR         1,<IRD>,<MIC>           ; IF NOT, SIGNAL AN ERROR
                        010C  1788          ENDLOOP         I                       ; CONTINUE THE LOOP
                        010F  1789
                        010F  1790          SUBTEST         3
                        0112
                        0112        ;////////////////////////////////////////////////////////////////////////////
                        0112  1791  ;+
                        0112  1792  ; Test the IRD1 ROMS with the FPA present and enabled.
                        0112  1793  ;-
                        0112  1794          LOADREG         WDREG,8$,,L             ; LOAD DATA TO ENABLE THE FPA
                        011A  1795          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
```

```
                    011C  1796          FETCH           <^X10>                  ; ENABLE THE FPA AND
                    0121  1797          BRSTCLK         <^X14>,INH              ; READ BACK FPA ENABLE
                    0125  1798          CMPREGMSK       CSTRP,4$,,5$,,W         ; IS THE FPA ENABLED?
                    0131  1799          IFERROR         ,,<FPA>                 ; IF NOT, SIGNAL AN ERROR
                    0137  1800          EXPUTRAP                                ; EXPECT NON-DCS ADDRESSES
                    0139  1801          LOOP            I,1,256                 ; LOOP FOR 256 OPCODES
                    0140  1802          SAVEINDEX       I,9$                    ; SAVE I-1 TO FORM AN OPCODE
                    0145  1803          LOADREG         WDREG,9$,,L             ; LOAD WDREG WITH THE OPCODE
                    014D  1804          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                    014F  1805          FETCH           0                       ; EXECUTE THE IRD1 ADDRESS TEST
                    0154  1806          BRSTCLK         <^X0F>,INH              ; ...
                    0158  1807          CMPREGMSK       CSTRP,11$,I,6$,,W       ; IS THE CS ADDRESS CORRECT?
                    0164  1808          IFERROR         1,<IRD>,<MIC>           ; IF NOT, SIGNAL AN ERROR
                    016B  1809          ENDLOOP         I                       ; CONTINUE THE LOOP
                    016E  1810          SKIP                                    ; SKIP TO THE NEXT TEST
                    0175  1811
                    0175  1812 BEGIN_TEST_DATA
                    0175
                    0175        ;-----------------------------------------------------------------------
                    0175  1813
                1802  0175  1814 1$:      .WORD   ^X1802          ; ADDRESS GENERATED WHEN FPA IS NOT PRESENT
                3FFE  0177  1815 2$:      .WORD   ^X3FFE          ; ADDRESS MASK FOR FPA PRESENCE TEST
                    0179  1816
                1801  0179  1817 3$:      .WORD   ^X1801          ; ADDRESS GENERATED WHEN THE FPA IS DISABLED
                1800  017B  1818 4$:      .WORD   ^X1800          ; ADDRESS GENERATED WHEN THE FPA IS ENABLED
                3FFD  017D  1819 5$:      .WORD   ^X3FFD          ; ADDRESS MASK FOR FPA ENABLE TEST
                    017F  1820
                3FFF  017F  1821 6$:      .WORD   ^X3FFF          ; ADDRESS COMPARISON MASK
                    0181  1822
            00000000  0181  1823 7$:      .LONG   0               ; DATA TO DISABLE THE FPA
            00000020  0185  1824 8$:      .LONG   ^X00000020      ; DATA TO ENABLE THE FPA
                    0189  1825
            00000000  0189  1826 9$:      .LONG   0               ; LOCATION TO HOLD LOOP COUNTER
                    018D  1827
                    018D  1828 ;
                    018D  1829 ; Expected CS Addresses for Subtests 1 and 2
                    018D  1830 ;
                    018D  1831 ; The following is a table of all the expected control store addresses for an
                    018D  1832 ; IRD1 with the FPA disabled. The hex numbers in the comments to the right
                    018D  1833 ; indicate the range of loop counter values for each row of address words.
                    018D  1834 ;
                    018D  1835
03A0 02A0 02B8 0308 03B8 0300 0310  018D  1836 10$:     .WORD   ^X0310,^X0300,^X03B8,^X0308,^X02B8,^X02A0,^X03A0 ; 01 - 07
0100 0100 0120 0100 0100 0100 03A8  019B  1837          .WORD   ^X03A8,^X0100,^X0100,^X0100,^X0120,^X0100,^X0100 ; 08 - 0E
0058 0058 0058 0078 01F8 0120 0120  01A9  1838          .WORD   ^X0120,^X0120,^X01F8,^X0078,^X0058,^X0058,^X0058 ; 0F - 15
0058 0058 0058 0058 0120 0120 0058  01B7  1839          .WORD   ^X0058,^X0120,^X0120,^X0058,^X0058,^X0058,^X0058 ; 16 - 1C
0100 0100 0100 0058 0058 0058 0058  01C5  1840          .WORD   ^X0058,^X0058,^X0058,^X0058,^X0100,^X0100,^X0100 ; 1D - 23
0100 0100 0100 0100 0100 0100 0100  01D3  1841          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; 24 - 2A
0218 0100 0100 0100 0100 0100 0100  01E1  1842          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0218 ; 2B - 31
0100 0100 0100 0100 0100 0100 0098  01EF  1843          .WORD   ^X0098,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; 32 - 38
0120 0100 0100 0100 0100 0100 0100  01FD  1844          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0120 ; 39 - 3F
0160 0160 0160 0160 0160 0160 0120  020B  1845          .WORD   ^X0120,^X0160,^X0160,^X0160,^X0160,^X0160,^X0160 ; 40 - 46
0100 0160 0160 0160 0160 0160 0160  0219  1846          .WORD   ^X0160,^X0160,^X0160,^X0160,^X0160,^X0160,^X0100 ; 47 - 4D
0160 0160 0160 0160 0160 0100 0100  0227  1847          .WORD   ^X0100,^X0100,^X0160,^X0160,^X0160,^X0160,^X0160 ; 4E - 54
03B0 03B0 0100 03B0 0160 0160 0160  0235  1848          .WORD   ^X0160,^X0160,^X0160,^X03B0,^X0100,^X03B0,^X03B0 ; 55 - 5B
```

```
 0180 0180 0120 0120 0120 0120 03B0   0243   1849          .WORD   ^X03B0,^X0120,^X0120,^X0120,^X0120,^X0180,^X0180 ; 5C - 62
 0180 0180 0180 0180 0180 0180 0180   0251   1850          .WORD   ^X0180,^X0180,^X0180,^X0180,^X0180,^X0180,^X0180 ; 63 - 69
 0180 0100 0100 0100 0180 0180 0180   025F   1851          .WORD   ^X0180,^X0180,^X0180,^X0100,^X0100,^X0100,^X0180 ; 6A - 70
 0180 0180 0180 0180 0180 0180 0180   026D   1852          .WORD   ^X0180,^X0180,^X0180,^X0180,^X0180,^X0180,^X0180 ; 71 - 77
 0170 0140 0100 0100 0100 0100 03B0   027B   1853          .WORD   ^X03B0,^X0100,^X0100,^X0100,^X0100,^X0140,^X0170 ; 78 - 7E
 0100 0100 0100 0100 0100 0120 0120   0289   1854          .WORD   ^X0120,^X0120,^X0100,^X0100,^X0100,^X0100,^X0100 ; 7F - 85
 0100 0100 0100 0100 0100 0100 0100   0297   1855          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; 86 - 8C
 0100 0100 0100 0100 0100 0100 0100   02A5   1856          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; 8D - 93
 0100 0100 0110 0110 0100 0140 0100   02B3   1857          .WORD   ^X0100,^X0140,^X0100,^X0110,^X0110,^X0100,^X0100 ; 94 - 9A
 0100 0120 0120 0100 0100 0100 0100   02C1   1858          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0120,^X0120,^X0100 ; 9B - A1
 0100 0100 0100 0100 0100 0100 0100   02CF   1859          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; A2 - A8
 0100 0100 0100 0100 0100 0100 0100   02DD   1860          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; A9 - AF
 0100 0140 0100 0100 0100 0100 0100   02EB   1861          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0140,^X0100 ; B0 - B6
 0100 0100 0100 0100 0100 0110 0110   02F9   1862          .WORD   ^X0110,^X0110,^X0100,^X0100,^X0100,^X0100,^X0100 ; B7 - BD
 0100 0100 0100 0100 0100 0100 0100   0307   1863          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; BE - C4
 0100 0100 0100 0100 0100 0100 0100   0315   1864          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; C5 - CB
 0100 0100 0100 0100 0100 0100 0100   0323   1865          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; CC - D2
 0100 0110 0110 0100 0140 0100 0100   0331   1866          .WORD   ^X0100,^X0100,^X0140,^X0100,^X0110,^X0110,^X0100 ; D3 - D9
 0120 0120 0100 0140 0100 0100 0100   033F   1867          .WORD   ^X0100,^X0100,^X0100,^X0140,^X0100,^X0120,^X0120 ; DA - E0
 0100 0100 0100 0100 0100 0100 0100   034D   1868          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; E1 - E7
 0100 0100 0100 0100 0100 0100 0100   035B   1869          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; E8 - EE
 01A0 0100 0100 0100 0100 0100 0100   0369   1870          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X01A0 ; EF - F5
 0100 0120 0100 0100 0100 0100 01A0   0377   1871          .WORD   ^X01A0,^X0100,^X0100,^X0100,^X0100,^X0120,^X0100 ; F6 - FC
                03B0 03B0 03C0 02C8   0385   1872          .WORD   ^X02C8,^X03C0,^X03B0,^X03B0                       ; FD - 100
                                      038D   1873
                                      038D   1874 ;
                                      038D   1875 ; Expected CS Addresses for Subtest 3
                                      038D   1876 ;
                                      038D   1877 ; The following is a table of all the expected control store addresses for an
                                      038D   1878 ; IRD1 with the FPA enabled. The hex numbers in the comments to the right
                                      038D   1879 ; indicate the range of loop counter values for each row of address words.
                                      038D   1880 ;
                                      038D   1881
 03A0 02A0 02B8 0308 03B8 0300 0310   038D   1882 11$:      .WORD   ^X0310,^X0300,^X03B8,^X0308,^X02B8,^X02A0,^X03A0 ; 01 - 07
 0100 0100 0120 0100 0100 0100 03A8   039B   1883          .WORD   ^X03A8,^X0100,^X0100,^X0100,^X0120,^X0100,^X0100 ; 08 - 0E
 0058 0058 0058 0078 01F8 0120 0120   03A9   1884          .WORD   ^X0120,^X0120,^X01F8,^X0078,^X0058,^X0058,^X0058 ; 0F - 15
 0058 0058 0058 0058 0120 0120 0058   03B7   1885          .WORD   ^X0058,^X0120,^X0120,^X0058,^X0058,^X0058,^X0058 ; 16 - 1C
 0100 0100 0100 0058 0058 0058 0058   03C5   1886          .WORD   ^X0058,^X0058,^X0058,^X0058,^X0100,^X0100,^X0100 ; 1D - 23
 0100 0100 0100 0100 0100 0100 0100   03D3   1887          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; 24 - 2A
 0218 0100 0100 0100 0100 0100 0100   03E1   1888          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0218 ; 2B - 31
 0100 0100 0100 0100 0100 0100 0098   03EF   1889          .WORD   ^X0098,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; 32 - 38
 0120 0100 0100 0100 0100 0100 0100   03FD   1890          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0120 ; 39 - 3F
 0100 0100 0100 0100 0100 0100 0120   040B   1891          .WORD   ^X0120,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; 40 - 46
 0100 0100 0100 0100 0100 0100 0100   0419   1892          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; 47 - 4D
 0160 0160 0100 0160 0160 0100 0100   0427   1893          .WORD   ^X0100,^X0100,^X0160,^X0160,^X0100,^X0160,^X0160 ; 4E - 54
 03B0 03B0 0100 03B0 0100 0160 0100   0435   1894          .WORD   ^X0100,^X0160,^X0100,^X03B0,^X0100,^X03B0,^X03B0 ; 55 - 5B
 01B0 01B0 0120 0120 0120 0120 03B0   0443   1895          .WORD   ^X03B0,^X0120,^X0120,^X0120,^X0120,^X01B0,^X01B0 ; 5C - 62
 01B0 01B0 01B0 01B0 01B0 01B0 01B0   0451   1896          .WORD   ^X01B0,^X01B0,^X01B0,^X01B0,^X01B0,^X01B0,^X01B0 ; 63 - 69
 0180 0100 0100 0100 01B0 01B0 01B0   045F   1897          .WORD   ^X01B0,^X01B0,^X01B0,^X0100,^X0100,^X0100,^X0180 ; 6A - 70
 01B0 01B0 01B0 0180 0180 01B0 0180   046D   1898          .WORD   ^X0180,^X01B0,^X0180,^X0180,^X01B0,^X01B0,^X01B0 ; 71 - 77
 0170 0140 0100 0100 0100 0100 03B0   047B   1899          .WORD   ^X03B0,^X0100,^X0100,^X0100,^X0100,^X0140,^X0170 ; 78 - 7E
 0100 0100 0100 0100 0100 0120 0120   0489   1900          .WORD   ^X0120,^X0120,^X0100,^X0100,^X0100,^X0100,^X0100 ; 7F - 85
 0100 0100 0100 0100 0100 0100 0100   0497   1901          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; 86 - 8C
 0100 0100 0100 0100 0100 0100 0100   04A5   1902          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; 8D - 93
 0100 0100 0110 0110 0100 0140 0100   04B3   1903          .WORD   ^X0100,^X0140,^X0100,^X0110,^X0110,^X0100,^X0100 ; 94 - 9A
```

```
 0100 0120 0120 0100 0100 0100 0100   04C1   1904          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0120,^X0120,^X0100 ; 9B - A1
 0100 0100 0100 0100 0100 0100 0100   04CF   1905          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; A2 - A8
 0100 0100 0100 0100 0100 0100 0100   04DD   1906          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; A9 - AF
 0100 0140 0100 0100 0100 0100 0100   04EB   1907          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0140,^X0100 ; B0 - B6
 0100 0100 0100 0100 0100 0110 0110   04F9   1908          .WORD   ^X0110,^X0110,^X0100,^X0100,^X0100,^X0100,^X0100 ; B7 - BD
 0100 0100 0100 0100 0100 0100 0100   0507   1909          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; BE - C4
 0100 0100 0100 0100 0100 0100 0100   0515   1910          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; C5 - CB
 0100 0100 0100 0100 0100 0100 0100   0523   1911          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; CC - D2
 0100 0110 0110 0100 0140 0100 0100   0531   1912          .WORD   ^X0100,^X0100,^X0140,^X0100,^X0110,^X0110,^X0100 ; D3 - D9
 0120 0120 0100 0140 0100 0100 0100   053F   1913          .WORD   ^X0100,^X0100,^X0100,^X0140,^X0100,^X0120,^X0120 ; DA - E0
 0100 0100 0100 0100 0100 0100 0100   054D   1914          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; E1 - E7
 0100 0100 0100 0100 0100 0100 0100   055B   1915          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X0100 ; E8 - EE
 01A0 0100 0100 0100 0100 0100 0100   0569   1916          .WORD   ^X0100,^X0100,^X0100,^X0100,^X0100,^X0100,^X01A0 ; EF - F5
 0100 0120 0100 0100 0100 0100 01A0   0577   1917          .WORD   ^X01A0,^X0100,^X0100,^X0100,^X0100,^X0120,^X0100 ; F6 - FC
                03B0 03B0 03C0 02C8   0585   1918          .WORD   ^X02C8,^X03C0,^X03B0,^X03B0                      ; FD - 100
                                      058D   1919
                                      058D   1920 BEGIN_CPU_DATA
                                      0002   1921
                                      0002   1922 DCS_DATA
                                      0001   1923
U 00, 7700.C800.0364.0300.0470.1801   0001   1924 ;% 00   NOP                                  ; GET UP TO SPEED
U 01, 5700.C840.0364.0304.0470.1802   000C   1928 ;% 01   R[TEMP1]_RDM                         ; LOAD THE OPCODE INTO RTEMP1
U 02, 7700.F800.57D7.D7FF.8470.1803   0017   1932 ;% 02   LONLIT_[50505000]                   ; LOAD THE OPERANDS INTO THE LONLIT REG
U 03, 7700.8800.5BE5.03D4.0470.1804   0022   1936 ;% 03   Q_R[LONLIT]                         ; MOVE THE OPERANDS TO THE Q REGISTER
U 04, 7700.0841.00A4.0304.0470.1805   002D   1940 ;% 04   R[TEMP1]_M[TEMP1].OR.Q              ; OR THE OPCODE WITH THE OPERANDS
U 05, 7700.4800.5BE4.03D8.4A70.1806   0038   1944 ;% 05   VA_R[ZERO]                          ; ZERO THE VA
U 06, 7701.C800.5BE4.0204.5590.1807   0043   1948 ;% 06   WRITE.LONG R[TEMP1]                 ; WRITE THE MODEL INSTRUCTION TO CACHE
U 07, 7700.8800.5BE4.0304.5470.1808   004E   1952 ;% 07   WDR_UNROT(R[TEMP1])               ; CMI OFF SO HOLD DATA FOR SECOND CYCLE
U 08, 7700.C800.0364.0300.4470.1809   0059   1956 ;% 08   VA_VA+4                           ; INCREMENT THE VA
U 09, 7701.0800.5BE4.02D8.5590.180A   0064   1960 ;% 09   WRITE.LONG R[ZERO]                 ; VALIDATE NEXT LOCATION FOR XB1
U 0A, 7700.C800.5BE4.03D8.5470.180B   006F   1964 ;% 0A   WDR_UNROT(R[ZERO])               ; CMI OFF SO HOLD DATA FOR SECOND CYCLE
U 0B, 7700.C800.5BE4.03D8.4870.180C   007A   1968 ;% 0B   PC_R[ZERO]                         ; START INSTRUCTION PREFETCH, LOAD XB'S
U 0C, 7701.0857.5924.0202.0470.180D   0085   1972 ;% 0C   R[TEMP0]_M[XB.PC_PC+I],ISIZE[LONG] ; LOAD RTEMP0 WITH XB0
U 0D, 7700.4800.5BE4.03D8.4870.180E   0090   1976 ;% 0D   PC_R[ZERO]                         ; RESET THE PC AND RELOAD THE XB'S
U 0E, 7700.0800.0364.1300.0470.180F   009B   1980 ;% 0E   BUT/IRD1                           ; EXECUTE AN INSTRUCTION DECODE
U 0F, 7300.C800.0364.0300.0470.1810   00A6   1984 ;% 0F   STOP.CLOCK                        ; HALT
                                      00B1   1988
U 10, 7700.4800.0364.0300.0470.1811   00B1   1989 ;% 10   NOP                                  ; GET CPU UP TO SPEED
U 11, 5700.4800.0364.0300.2A70.1812   00BC   1993 ;% 11   WCTRL/FPA.ENABLE_WB5,WB_RDM         ; WRITE THE FPA'S ENABLE BIT
U 12, 7700.C800.0364.0300.0470.1813   00C7   1997 ;% 12   NOP                                  ; NOP TO GET CPU UP TO SPEED
U 13, 7700.4800.0364.C300.0470.1800   00D2   2001 ;% 13   BUT/NO.FPA,NEXT/1800               ; CHECK FPA STATUS
U 14, 7300.C800.0364.0300.0470.1815   00DD   2005 ;% 14   STOP.CLOCK                        ; HALT
                                      00E8   2009
U 15, 7700.C800.0364.0300.0470.1816   00E8   2010 ;% 15   NOP                                  ; NOP TO GET CPU UP TO SPEED
U 16, 7700.F800.7F7F.7F7F.0470.1817   00F3   2014 ;% 16   LONLIT_[01010101]                   ; LOAD LONGLIT WITH 4 NOP INSTRUCTIONS
U 17, 7700.4800.5BE4.03D8.4A70.1818   00FE   2018 ;% 17   VA_R[ZERO]                          ; ZERO THE VA
U 18, 7701.8800.5BE4.02D4.5590.1819   0109   2022 ;% 18   WRITE.LONG R[LONLIT]               ; WRITE NOPS TO CACHE FOR XB0
U 19, 7700.C800.5BE4.03D4.5470.181A   0114   2026 ;% 19   WDR_UNROT(R[LONLIT])             ; CMI OFF SO HOLD FOR SECOND CYCLE
U 1A, 7700.C800.0364.0300.4470.181B   011F   2030 ;% 1A   VA_VA+4                           ; INCREMENT THE VA
U 1B, 7701.8800.5BE4.02D4.5590.181C   012A   2034 ;% 1B   WRITE.LONG R[LONLIT]               ; VALIDATE NEXT LOCATION FOR XB1
U 1C, 7700.4800.5BE4.03D4.5470.181D   0135   2038 ;% 1C   WDR_UNROT(R[LONLIT])             ; CMI OFF SO HOLD FOR SECOND CYCLE
U 1D, 7700.C800.5BE4.03D8.4870.181E   0140   2042 ;% 1D   PC_R[ZERO]                         ; ZERO THE PC TO LOAD XB'S
U 1E, 7700.C800.0364.1700.0470.181F   014B   2046 ;% 1E   IRD1TEST                          ; IRD1 TO SET FPA FOR NON-FPA OPCODE
U 1F, 7300.C800.0364.0300.0470.1820   0156   2050 ;% 1F   STOP.CLOCK                        ; HALT
                                      0161   2054
```

```
ZZ-ECKAB-9.2    "TEST 0C2 IRD1 Instruction Address Decod          13-MAY-1987      Fiche 5 Frame E12        Sequence 971
ECKAB                      VAX-11/750 DPM MICRO DIAGNOSTICS        11-DEC-1985 09:34:58  VAX/VMS Macro V04-00        Page  6
V09.01                     "TEST 0C2 IRD1 Instruction Address Decod 11-DEC-1985 09:34:05  [VAX750.DPM]ECKAB5.MAR;1          (1

                            0161  2055 RTEMP_DATA
                            0001  2056
                  00000000  0001  2057          .LONG   0       ; RTEMP0 HOLDS THE CONTENTS OF XB0
                  00000000  0005  2058          .LONG   0       ; RTEMP1 HOLDS THE INSTRUCTION OPCODE
                            0009  2059
                            0009  2060 END_CPU_DATA
                            058D  2061
                            058D  2062 END_TEST_DATA
                            058D
                            058D        ;-------------------------------------------------------------------------
                            058D  2063
                            058D  2064 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
                        002B        .SBTTL  "TEST   0C3     IRD1 Instruction Address Decode Test 2
                        002B  2066 ;*******************************************************************************
                        002B  2067 ;++
                        002B  2068 ;
                        002B  2069 ; FUNCTIONAL DESCRIPTION
                        002B  2070 ;
                        002B  2071 ;       This test will examine the signal ROM OS INH H from the ROMS that
                        002B  2072 ;       implement a native mode IRD1 when the First Part Done (FPD) bit of the
                        002B  2073 ;       PSL is set to zero.
                        002B  2074 ;
                        002B  2075 ; TEST ALGORITHM
                        002B  2076 ;
                        002B  2077 ;       This test is implemented in three subtests. The first subtest examines
                        002B  2078 ;       the IRD1 ROMS when the CPU's Floating Point Accelerator (FPA) is not
                        002B  2079 ;       present. The second subtest examines the ROMS when there is an FPA
                        002B  2080 ;       present and it has been disabled. The third subtest examines the ROMS
                        002B  2081 ;       when the FPA is present and enabled. The test algorithm executes as
                        002B  2082 ;       follows:
                        002B  2083 ;
                        002B  2084 ;        1. Initialize the CPU. This will clear the FPD bit in the PSL.
                        002B  2085 ;        2. Execute the following microcode to initialize an FPA:
                        002B  2086 ;                1. NOP to get the CPU up to speed.
                        002B  2087 ;                2. Load four NOP opcodes into the longlit register for XB0.
                        002B  2088 ;                3. Zero the VA.
                        002B  2089 ;                4. Write the NOPS for XB0 into cache at address 0.
                        002B  2090 ;                5. Increment the VA by 4.
                        002B  2091 ;                6. Validate the next location for XB1 prefetch by writing
                        002B  2092 ;                   NOPS to cache at address 4.
                        002B  2093 ;                7. Zero the PC to initiate the instruction prefetch that
                        002B  2094 ;                   writes XB0 and XB1 from cache.
                        002B  2095 ;                8. Execute an IRD1TEST cycle to decode the NOP in XB0.
                        002B  2096 ;                   This will prepare an FPA to be enabled or disabled.
                        002B  2097 ;                9. Halt.
                        002B  2098 ;        3. Execute the following microcode to test for an FPA presence:
                        002B  2099 ;               10. NOP to get the CPU up to speed.
                        002B  2100 ;               11. BUT/NO.FPA,NEXT/1800
                        002B  2101 ;               12. STOP.CLOCK to halt.
                        002B  2102 ;        4. Perform a masked comparison of the address in the Control Store
                        002B  2103 ;           Address Trace Register (CSTRP) against the expected address for the
                        002B  2104 ;           FPA not present.
                        002B  2105 ;        5. If the two do not match, then there is an FPA so branch to
                        002B  2106 ;           Subtest 2. Otherwise continue on to Subtest 1.
                        002B  2107 ;
                        002B  2108 ;       Subtest 1
                        002B  2109 ;
                        002B  2110 ;        6. Loop on error from here.
                        002B  2111 ;        7. Re-execute the microcode in step 3 to test the FPA enable bit.
                        002B  2112 ;        8. Perform a masked comparison of the CSTRP with the expected address
                        002B  2113 ;           for the FPA disabled.
                        002B  2114 ;        9. If the two do not match then there is a problem with the FPA/CPU
                        002B  2115 ;           interface so signal an error.
                        002B  2116 ;       10. Prepare the RDM to expect non-DCS addresses.
                        002B  2117 ;       11. Prepare a loop on I for 256 VAX opcodes.
                        002B  2118 ;       12. Save the loop counter minus one in the RDM memory.
                        002B  2119 ;       13. Load the WDREG with the number saved in RDM memory.
```

```
002B  2120 ;       14. Loop on error from here.
002B  2121 ;       15. Execute the following test microcode:
002B  2122 ;                13. NOP to get the CPU up to speed.
002B  2123 ;                14. Load the opcode to be tested from the WDREG to RTEMP1.
002B  2124 ;                15. Load the operands into the longlit register.
002B  2125 ;                16. Move the operands from the longlit register into the
002B  2126 ;                    Q register.
002B  2127 ;                17. Form the test instruction by ORing the opcode in RTEMP1
002B  2128 ;                    (MTEMP1=RTEMP1) with the operands in the Q register and
002B  2129 ;                    write the result back to RTEMP1.
002B  2130 ;                18. Zero the VA.
002B  2131 ;                19. Write the test instruction in RTEMP1 to cache at address 0.
002B  2132 ;                20. Increment the VA by 4.
002B  2133 ;                21. Validate the next location for XB1 prefetch by writing
002B  2134 ;                    zeros to cache at address 4.
002B  2135 ;                22. Zero the PC to initiate the instruction prefetch that
002B  2136 ;                    writes XB0 and XB1 from cache.
002B  2137 ;                23. Write the contents of XB0 to RTEMP0. This will cause the
002B  2138 ;                    PC to increment by 4 and can change the contents of XB0.
002B  2139 ;                24. Rezero the PC to reload XB0 and XB1.
002B  2140 ;                25. Execute a BUT/IRD1 to decode the instruction in XB0 and
002B  2141 ;                    strobe the VBUS.
002B  2142 ;                26. Halt.
002B  2143 ;       16. Compare the VBUS bit 1E (LOAD OSR L) with the expected value.
002B  2144 ;       17. If the two are not equal, signal an error.
002B  2145 ;       18. Continue to loop on I for the remaining VAX opcodes.
002B  2146 ;       19. Skip to the next test.
002B  2147 ;
002B  2148 ;       Subtests 2 and 3
002B  2149 ;
002B  2150 ;       20. Load the WDREG with a longword to perform an FPA disable for
002B  2151 ;           Subtest 2 or an FPA enable for Subtest 3.
002B  2152 ;       21. Loop on error from here.
002B  2153 ;       22. Execute the following microcode to disable the FPA for Subtest 2 or
002B  2154 ;           to enable the FPA for Subtest 3 and to also check the result:
002B  2155 ;                26. NOP to get the CPU up to speed.
002B  2156 ;                27. WCTRL/FPAENABLE_WB5,WB_RDM to write the FPA's enable bit
002B  2157 ;                    with bit 5 of the RDM's WDREG.
002B  2158 ;                28. NOP
002B  2159 ;                29. BUT/NO.FPA,NEXT/1800 to test for FPA enabled.
002B  2160 ;                30. STOP.CLOCK to halt.
002B  2161 ;       23. Perform a masked comparison of the CSTRP with the expected address
002B  2162 ;           for the FPA disabled (Subtest 2) or the FPA enabled (Subtest 3).
002B  2163 ;       24. If the two do not match then there is a problem with the FPA/CPU
002B  2164 ;           interface so signal an error.
002B  2165 ;       25. Prepare the RDM to expect non-DCS addresses.
002B  2166 ;       26. Prepare a loop on I for 256 VAX opcodes.
002B  2167 ;       27. Save the loop counter minus one in the RDM memory.
002B  2168 ;       28. Load the WDREG with the number saved in RDM memory.
002B  2169 ;       29. Loop on error from here.
002B  2170 ;       30. Execute the same test microcode as in Subtest 1 step 15.
002B  2171 ;       31. Compare the VBUS bit 1E (LOAD OSR L) with the expected value.
002B  2172 ;       32. If the two are not equal, signal an error.
002B  2173 ;       33. Continue to loop on I for the remaining VAX opcodes.
002B  2174 ;       34. If this is Subtest 2 then continue on to Subtest 3.
```

```
                        002B  2175 ;           If this is Subtest 3 then skip to the next test.
                        002B  2176 ;
                        002B  2177 ; TEST PATTERNS
                        002B  2178 ;
                        002B  2179 ;       For this test the VAX machine opcodes are generated by a simple count
                        002B  2180 ;       pattern. The opcodes are based on the 256 base machine instructions
                        002B  2181 ;       as specified in the VAX architecture handbook. The generated opcodes
                        002B  2182 ;       are written to the CPU and combined with three bytes of operand
                        002B  2183 ;       data to form a VAX machine instruction. The operands all specify R0 in
                        002B  2184 ;       register mode and are loaded from the CPU's long literal register.
                        002B  2185 ;
                        002B  2186 ;       Also required are two tables of expected VBUS values in bit 1E for
                        002B  2187 ;       each IRD1 Instruction Decode Cycle. One table contains the expected
                        002B  2188 ;       VBUS data for an IRD1 when the FPA has been disabled, the other for
                        002B  2189 ;       when it is enabled.  The expected VBUS data was obtained from the
                        002B  2190 ;       Native Mode Ird Rom Definition  in the CPU microcode.
                        002B  2191 ;
                        002B  2192 ; LOGIC DESCRIPTION
                        002B  2193 ;
                        002B  2194 ;       The CCS ROMS are fed addresses from ROMS on the DPM during instruction
                        002B  2195 ;       decode. The DPM ROMS are addressed by bits <07:00> of the execution
                        002B  2196 ;       buffer. These bits form the opcode for the instruction. For zero and
                        002B  2197 ;       certain single operand instructions the addresses generated by the ROMS
                        002B  2198 ;       will force a jump to execution flows in the microcode. For the rest,
                        002B  2199 ;       operand specifier evaulation addresses will be generated.
                        002B  2200 ;
                        002B  2201 ; ASSUMPTIONS
                        002B  2202 ;
                        002B  2203 ;       This test assumes the following logic has already been tested; the
                        002B  2204 ;       CPU's data path, the cache, the XB, and the IR and OSR.
                        002B  2205 ;
                        002B  2206 ; ERROR DESCRIPTION
                        002B  2207 ;
                        002B  2208 ;       On error this test can type out the following information:
                        002B  2209 ;
                        002B  2210 ;       Failing gate arrays:    IRD
                        002B  2211 ;       Failing modules:        FPA, MIC
                        002B  2212 ;
                        002B  2213 ;       The following data will also by typed:
                        002B  2214 ;
                        002B  2215 ;       Expected VBUS value 00000xnn            where   x  = VBUS bit value
                        002B  2216 ;       Received VBUS value 00000xnn                    nn = VBUS bit position
                        002B  2217 ;       Loop count (index I)
                        002B  2218 ;       The instruction being decoded from XB0 (RTEMP0)
                        002B  2219 ;
                        002B  2220 ;--
                        002B  2221 ;*******************************************************************************
                        002B  2222
                        002B  2223          INITIALIZE                              ; INIT THE CPU AND CLEAR FPD
                        002D  2224          FETCH           <^X15>                  ; PREPARE AN FPA TO BE ENABLED
                        0032  2225          BRSTCLK         <^X1F>                  ; OR DISABLED
                        0036  2226          FETCH           <^X12>                  ; EXECUTE A BUT/NO.FPA TO TEST
                        003B  2227          BRSTCLK         <^X14>                  ; FOR AN FPA PRESENT
                        003F  2228          CMPREGMSK       CSTRP,1$,,2$,,W         ; CHECK THE BUT/NO.FPA ADDRESS
                        004B  2229          SKIP            100$,ONERROR            ; SKIP IF THE FPA IS PRESENT
```

```
                        0052  2230
                        0052  2231          SUBTEST         1
                        0055
                        0055          ;////////////////////////////////////////////////////////////////////////////////
                        0055  2232 ;+
                        0055  2233 ; Test the IRD1 ROMS when the FPA is not present.
                        0055  2234 ;-
                        0055  2235          ERRLOOP                                 ; LOOP ON ERROR HERE
                        0057  2236          FETCH           <^X12>                  ; EXECUTE A BUT/NO.FPA TO TEST
                        005C  2237          BRSTCLK         <^X14>,INH              ; FOR THE FPA DISABLED
                        0060  2238          CMPREGMSK       CSTRP,3$,,5$,,W         ; IS THE FPA DISABLED?
                        006C  2239          IFERROR         ,,<FPA>                 ; IF NOT, SIGNAL AN ERROR
                        0072  2240          EXPUTRAP                                ; ELSE EXPECT NON-DCS ADDRESSES
                        0074  2241          LOOP            I,1,256                 ; LOOP FOR 256 OPCODES
                        007B  2242          SAVEINDEX       I,9$                    ; SAVE I-1 TO FORM AN OPCODE
                        0080  2243          LOADREG         WDREG,9$,,L             ; LOAD WDREG WITH THE OPCODE
                        0088  2244          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        008A  2245          FETCH           0                       ; EXECUTE THE IRD1 ADDRESS TEST
                        008F  2246          BRSTCLK         <^X0F>,INH              ; ...
                        0093  2247          CMPVBUS         10$,I                   ; IS VBUS BIT 1E CORRECT?
                        0098  2248          IFERROR         1,<IRD>,<MIC>           ; IF NOT, SIGNAL AN ERROR
                        009F  2249          ENDLOOP         I                       ; CONTINUE THE LOOP
                        00A2  2250          SKIP                                    ; SKIP TO THE NEXT TEST
                        00A9  2251
                        00A9  2252 100$:    SUBTEST         2
                        00AC
                        00AC          ;////////////////////////////////////////////////////////////////////////////////
                        00AC  2253 ;+
                        00AC  2254 ; Test the IRD1 ROMS with the FPA present but not enabled.
                        00AC  2255 ;-
                        00AC  2256          LOADREG         WDREG,7$,,L             ; LOAD DATA TO DISABLE THE FPA
                        00B4  2257          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        00B6  2258          FETCH           <^X10>                  ; EXECUTE AN FPA DISABLE AND
                        00BB  2259          BRSTCLK         <^X14>,INH              ; READ BACK THE FPA STATUS
                        00BF  2260          CMPREGMSK       CSTRP,3$,,5$,,W         ; IS THE FPA DISABLED?
                        00CB  2261          IFERROR         ,,<FPA>                 ; IF NOT, SIGNAL AN ERROR
                        00D1  2262          EXPUTRAP                                ; EXPECT NON-DCS ADDRESSES
                        00D3  2263          LOOP            I,1,256                 ; LOOP FOR 256 OPCODES
                        00DA  2264          SAVEINDEX       I,9$                    ; SAVE I-1 TO FORM AN OPCODE
                        00DF  2265          LOADREG         WDREG,9$,,L             ; LOAD WDREG WITH THE OPCODE
                        00E7  2266          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        00E9  2267          FETCH           0                       ; EXECUTE THE IRD1 ADDRESS TEST
                        00EE  2268          BRSTCLK         <^X0F>,INH              ; ...
                        00F2  2269          CMPVBUS         10$,I                   ; IS VBUS BIT 1E CORRECT?
                        00F7  2270          IFERROR         1,<IRD>,<MIC>           ; IF NOT, SIGNAL AN ERROR
                        00FE  2271          ENDLOOP         I                       ; CONTINUE THE LOOP
                        0101  2272
                        0101  2273          SUBTEST         3
                        0104
                        0104          ;////////////////////////////////////////////////////////////////////////////////
                        0104  2274 ;+
                        0104  2275 ; Test the IRD1 ROMS with the FPA present and enabled.
                        0104  2276 ;-
                        0104  2277          LOADREG         WDREG,8$,,L             ; LOAD DATA TO ENABLE THE FPA
                        010C  2278          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
```

```
                    010E  2279          FETCH           <^X10>                  ; ENABLE THE FPA AND
                    0113  2280          BRSTCLK         <^X14>,INH              ; READ BACK FPA ENABLE
                    0117  2281          CMPREGMSK       CSTRP,4$,,5$,,W         ; IS THE FPA ENABLED?
                    0123  2282          IFERROR         ,,<FPA>                 ; IF NOT, SIGNAL AN ERROR
                    0129  2283          EXPUTRAP                                ; EXPECT NON-DCS ADDRESSES
                    012B  2284          LOOP            I,1,256                 ; LOOP FOR 256 OPCODES
                    0132  2285          SAVEINDEX       I,9$                    ; SAVE I-1 TO FORM AN OPCODE
                    0137  2286          LOADREG         WDREG,9$,,L             ; LOAD WDREG WITH THE OPCODE
                    013F  2287          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                    0141  2288          FETCH           0                       ; EXECUTE THE IRD1 ADDRESS TEST
                    0146  2289          BRSTCLK         <^X0F>,INH              ; ...
                    014A  2290          CMPVBUS         11$,I                   ; IS VBUS BIT 1E CORRECT?
                    014F  2291          IFERROR         1,<IRD>,<MIC>           ; IF NOT, SIGNAL AN ERROR
                    0156  2292          ENDLOOP         I                       ; CONTINUE THE LOOP
                    0159  2293          SKIP                                    ; SKIP TO THE NEXT TEST
                    0160  2294
                    0160  2295 BEGIN_TEST_DATA
                    0160
                    0160        ;-------------------------------------------------------------------------
                    0160  2296
                1802 0160  2297 1$:     .WORD   ^X1802          ; ADDRESS GENERATED WHEN FPA IS NOT PRESENT
                3FFE 0162  2298 2$:     .WORD   ^X3FFE          ; ADDRESS MASK FOR FPA PRESENCE TEST
                    0164  2299
                1801 0164  2300 3$:     .WORD   ^X1801          ; ADDRESS GENERATED WHEN THE FPA IS DISABLED
                1800 0166  2301 4$:     .WORD   ^X1800          ; ADDRESS GENERATED WHEN THE FPA IS ENABLED
                3FFD 0168  2302 5$:     .WORD   ^X3FFD          ; ADDRESS MASK FOR FPA ENABLE TEST
                    016A  2303
                3FFF 016A  2304 6$:     .WORD   ^X3FFF          ; ADDRESS COMPARISON MASK
                    016C  2305
            00000000 016C  2306 7$:     .LONG   0               ; DATA TO DISABLE THE FPA
            00000020 0170  2307 8$:     .LONG   ^X00000020      ; DATA TO ENABLE THE FPA
                    0174  2308
            00000000 0174  2309 9$:     .LONG   0               ; LOCATION TO HOLD LOOP COUNTER
                    0178  2310
                    0178  2311 ;
                    0178  2312 ; Expected VBUS bit values for Subtests 1 and 2
                    0178  2313 ;
                    0178  2314 ; The following is a table of all the VBUS bit values for an IRD1 with the FPA
                    0178  2315 ; disabled. The hex numbers in the comments to the right indicate the range of
                    0178  2316 ; loop counter values for each row of address words.
                    0178  2317 ;
                    0178  2318
                  01 0178  2319 10$:    .BYTE   ^X01
                    0179  2320          VBUSDATA        <^X1E>,1                ;00
                  01 017A  2321          .BYTE   ^X01
                    017B  2322          VBUSDATA        <^X1E>,1                ;01
                  01 017C  2323          .BYTE   ^X01
                    017D  2324          VBUSDATA        <^X1E>,1                ;02
                  01 017E  2325          .BYTE   ^X01
                    017F  2326          VBUSDATA        <^X1E>,1                ;03
                  01 0180  2327          .BYTE   ^X01
                    0181  2328          VBUSDATA        <^X1E>,1                ;04
                  01 0182  2329          .BYTE   ^X01
                    0183  2330          VBUSDATA        <^X1E>,1                ;05
                  01 0184  2331          .BYTE   ^X01
```

```
                0185  2332          VBUSDATA        <^X1E>,1                ;06
    01          0186  2333          .BYTE   ^X01
                0187  2334          VBUSDATA        <^X1E>,1                ;07
    01          0188  2335          .BYTE   ^X01
                0189  2336          VBUSDATA        <^X1E>,0                ;08
    01          018A  2337          .BYTE   ^X01
                018B  2338          VBUSDATA        <^X1E>,0                ;09
    01          018C  2339          .BYTE   ^X01
                018D  2340          VBUSDATA        <^X1E>,0                ;0A
    01          018E  2341          .BYTE   ^X01
                018F  2342          VBUSDATA        <^X1E>,0                ;0B
    01          0190  2343          .BYTE   ^X01
                0191  2344          VBUSDATA        <^X1E>,0                ;0C
    01          0192  2345          .BYTE   ^X01
                0193  2346          VBUSDATA        <^X1E>,0                ;0D
    01          0194  2347          .BYTE   ^X01
                0195  2348          VBUSDATA        <^X1E>,0                ;0E
    01          0196  2349          .BYTE   ^X01
                0197  2350          VBUSDATA        <^X1E>,0                ;0F
    01          0198  2351          .BYTE   ^X01
                0199  2352          VBUSDATA        <^X1E>,0                ;10
    01          019A  2353          .BYTE   ^X01
                019B  2354          VBUSDATA        <^X1E>,0                ;11
    01          019C  2355          .BYTE   ^X01
                019D  2356          VBUSDATA        <^X1E>,0                ;12
    01          019E  2357          .BYTE   ^X01
                019F  2358          VBUSDATA        <^X1E>,0                ;13
    01          01A0  2359          .BYTE   ^X01
                01A1  2360          VBUSDATA        <^X1E>,0                ;14
    01          01A2  2361          .BYTE   ^X01
                01A3  2362          VBUSDATA        <^X1E>,0                ;15
    01          01A4  2363          .BYTE   ^X01
                01A5  2364          VBUSDATA        <^X1E>,0                ;16
    01          01A6  2365          .BYTE   ^X01
                01A7  2366          VBUSDATA        <^X1E>,0                ;17
    01          01A8  2367          .BYTE   ^X01
                01A9  2368          VBUSDATA        <^X1E>,0                ;18
    01          01AA  2369          .BYTE   ^X01
                01AB  2370          VBUSDATA        <^X1E>,0                ;19
    01          01AC  2371          .BYTE   ^X01
                01AD  2372          VBUSDATA        <^X1E>,0                ;1A
    01          01AE  2373          .BYTE   ^X01
                01AF  2374          VBUSDATA        <^X1E>,0                ;1B
    01          01B0  2375          .BYTE   ^X01
                01B1  2376          VBUSDATA        <^X1E>,0                ;1C
    01          01B2  2377          .BYTE   ^X01
                01B3  2378          VBUSDATA        <^X1E>,0                ;1D
    01          01B4  2379          .BYTE   ^X01
                01B5  2380          VBUSDATA        <^X1E>,0                ;1E
    01          01B6  2381          .BYTE   ^X01
                01B7  2382          VBUSDATA        <^X1E>,0                ;1F
    01          01B8  2383          .BYTE   ^X01
                01B9  2384          VBUSDATA        <^X1E>,0                ;20
    01          01BA  2385          .BYTE   ^X01
                01BB  2386          VBUSDATA        <^X1E>,0                ;21
```

```
                        01  01BC  2387          .BYTE   ^X01
                            01BD  2388          VBUSDATA        <^X1E>,0                ;22
                        01  01BE  2389          .BYTE   ^X01
                            01BF  2390          VBUSDATA        <^X1E>,0                ;23
                        01  01C0  2391          .BYTE   ^X01
                            01C1  2392          VBUSDATA        <^X1E>,0                ;24
                        01  01C2  2393          .BYTE   ^X01
                            01C3  2394          VBUSDATA        <^X1E>,0                ;25
                        01  01C4  2395          .BYTE   ^X01
                            01C5  2396          VBUSDATA        <^X1E>,0                ;26
                        01  01C6  2397          .BYTE   ^X01
                            01C7  2398          VBUSDATA        <^X1E>,0                ;27
                        01  01C8  2399          .BYTE   ^X01
                            01C9  2400          VBUSDATA        <^X1E>,0                ;28
                        01  01CA  2401          .BYTE   ^X01
                            01CB  2402          VBUSDATA        <^X1E>,0                ;29
                        01  01CC  2403          .BYTE   ^X01
                            01CD  2404          VBUSDATA        <^X1E>,0                ;2A
                        01  01CE  2405          .BYTE   ^X01
                            01CF  2406          VBUSDATA        <^X1E>,0                ;2B
                        01  01D0  2407          .BYTE   ^X01
                            01D1  2408          VBUSDATA        <^X1E>,0                ;2C
                        01  01D2  2409          .BYTE   ^X01
                            01D3  2410          VBUSDATA        <^X1E>,0                ;2D
                        01  01D4  2411          .BYTE   ^X01
                            01D5  2412          VBUSDATA        <^X1E>,0                ;2E
                        01  01D6  2413          .BYTE   ^X01
                            01D7  2414          VBUSDATA        <^X1E>,0                ;2F
                        01  01D8  2415          .BYTE   ^X01
                            01D9  2416          VBUSDATA        <^X1E>,0                ;30
                        01  01DA  2417          .BYTE   ^X01
                            01DB  2418          VBUSDATA        <^X1E>,0                ;31
                        01  01DC  2419          .BYTE   ^X01
                            01DD  2420          VBUSDATA        <^X1E>,0                ;32
                        01  01DE  2421          .BYTE   ^X01
                            01DF  2422          VBUSDATA        <^X1E>,0                ;33
                        01  01E0  2423          .BYTE   ^X01
                            01E1  2424          VBUSDATA        <^X1E>,0                ;34
                        01  01E2  2425          .BYTE   ^X01
                            01E3  2426          VBUSDATA        <^X1E>,0                ;35
                        01  01E4  2427          .BYTE   ^X01
                            01E5  2428          VBUSDATA        <^X1E>,0                ;36
                        01  01E6  2429          .BYTE   ^X01
                            01E7  2430          VBUSDATA        <^X1E>,0                ;37
                        01  01E8  2431          .BYTE   ^X01
                            01E9  2432          VBUSDATA        <^X1E>,0                ;38
                        01  01EA  2433          .BYTE   ^X01
                            01EB  2434          VBUSDATA        <^X1E>,0                ;39
                        01  01EC  2435          .BYTE   ^X01
                            01ED  2436          VBUSDATA        <^X1E>,0                ;3A
                        01  01EE  2437          .BYTE   ^X01
                            01EF  2438          VBUSDATA        <^X1E>,0                ;3B
                        01  01F0  2439          .BYTE   ^X01
                            01F1  2440          VBUSDATA        <^X1E>,0                ;3C
                        01  01F2  2441          .BYTE   ^X01
```

```
                    01F3  2442          VBUSDATA        <^X1E>,0                ;3D
                01  01F4  2443          .BYTE   ^X01
                    01F5  2444          VBUSDATA        <^X1E>,0                ;3E
                01  01F6  2445          .BYTE   ^X01
                    01F7  2446          VBUSDATA        <^X1E>,0                ;3F
                01  01F8  2447          .BYTE   ^X01
                    01F9  2448          VBUSDATA        <^X1E>,0                ;40
                01  01FA  2449          .BYTE   ^X01
                    01FB  2450          VBUSDATA        <^X1E>,0                ;41
                01  01FC  2451          .BYTE   ^X01
                    01FD  2452          VBUSDATA        <^X1E>,0                ;42
                01  01FE  2453          .BYTE   ^X01
                    01FF  2454          VBUSDATA        <^X1E>,0                ;43
                01  0200  2455          .BYTE   ^X01
                    0201  2456          VBUSDATA        <^X1E>,0                ;44
                01  0202  2457          .BYTE   ^X01
                    0203  2458          VBUSDATA        <^X1E>,0                ;45
                01  0204  2459          .BYTE   ^X01
                    0205  2460          VBUSDATA        <^X1E>,0                ;46
                01  0206  2461          .BYTE   ^X01
                    0207  2462          VBUSDATA        <^X1E>,0                ;47
                01  0208  2463          .BYTE   ^X01
                    0209  2464          VBUSDATA        <^X1E>,0                ;48
                01  020A  2465          .BYTE   ^X01
                    020B  2466          VBUSDATA        <^X1E>,0                ;49
                01  020C  2467          .BYTE   ^X01
                    020D  2468          VBUSDATA        <^X1E>,0                ;4A
                01  020E  2469          .BYTE   ^X01
                    020F  2470          VBUSDATA        <^X1E>,0                ;4B
                01  0210  2471          .BYTE   ^X01
                    0211  2472          VBUSDATA        <^X1E>,0                ;4C
                01  0212  2473          .BYTE   ^X01
                    0213  2474          VBUSDATA        <^X1E>,0                ;4D
                01  0214  2475          .BYTE   ^X01
                    0215  2476          VBUSDATA        <^X1E>,0                ;4E
                01  0216  2477          .BYTE   ^X01
                    0217  2478          VBUSDATA        <^X1E>,0                ;4F
                01  0218  2479          .BYTE   ^X01
                    0219  2480          VBUSDATA        <^X1E>,0                ;50
                01  021A  2481          .BYTE   ^X01
                    021B  2482          VBUSDATA        <^X1E>,0                ;51
                01  021C  2483          .BYTE   ^X01
                    021D  2484          VBUSDATA        <^X1E>,0                ;52
                01  021E  2485          .BYTE   ^X01
                    021F  2486          VBUSDATA        <^X1E>,0                ;53
                01  0220  2487          .BYTE   ^X01
                    0221  2488          VBUSDATA        <^X1E>,0                ;54
                01  0222  2489          .BYTE   ^X01
                    0223  2490          VBUSDATA        <^X1E>,0                ;55
                01  0224  2491          .BYTE   ^X01
                    0225  2492          VBUSDATA        <^X1E>,0                ;56
                01  0226  2493          .BYTE   ^X01
                    0227  2494          VBUSDATA        <^X1E>,1                ;57
                01  0228  2495          .BYTE   ^X01
                    0229  2496          VBUSDATA        <^X1E>,0                ;58
```

```
                    01  022A  2497          .BYTE   ^X01
                        022B  2498          VBUSDATA        <^X1E>,1                ;59
                    01  022C  2499          .BYTE   ^X01
                        022D  2500          VBUSDATA        <^X1E>,1                ;5A
                    01  022E  2501          .BYTE   ^X01
                        022F  2502          VBUSDATA        <^X1E>,1                ;5B
                    01  0230  2503          .BYTE   ^X01
                        0231  2504          VBUSDATA        <^X1E>,0                ;5C
                    01  0232  2505          .BYTE   ^X01
                        0233  2506          VBUSDATA        <^X1E>,0                ;5D
                    01  0234  2507          .BYTE   ^X01
                        0235  2508          VBUSDATA        <^X1E>,0                ;5E
                    01  0236  2509          .BYTE   ^X01
                        0237  2510          VBUSDATA        <^X1E>,0                ;5F
                    01  0238  2511          .BYTE   ^X01
                        0239  2512          VBUSDATA        <^X1E>,0                ;60
                    01  023A  2513          .BYTE   ^X01
                        023B  2514          VBUSDATA        <^X1E>,0                ;61
                    01  023C  2515          .BYTE   ^X01
                        023D  2516          VBUSDATA        <^X1E>,0                ;62
                    01  023E  2517          .BYTE   ^X01
                        023F  2518          VBUSDATA        <^X1E>,0                ;63
                    01  0240  2519          .BYTE   ^X01
                        0241  2520          VBUSDATA        <^X1E>,0                ;64
                    01  0242  2521          .BYTE   ^X01
                        0243  2522          VBUSDATA        <^X1E>,0                ;65
                    01  0244  2523          .BYTE   ^X01
                        0245  2524          VBUSDATA        <^X1E>,0                ;66
                    01  0246  2525          .BYTE   ^X01
                        0247  2526          VBUSDATA        <^X1E>,0                ;67
                    01  0248  2527          .BYTE   ^X01
                        0249  2528          VBUSDATA        <^X1E>,0                ;68
                    01  024A  2529          .BYTE   ^X01
                        024B  2530          VBUSDATA        <^X1E>,0                ;69
                    01  024C  2531          .BYTE   ^X01
                        024D  2532          VBUSDATA        <^X1E>,0                ;6A
                    01  024E  2533          .BYTE   ^X01
                        024F  2534          VBUSDATA        <^X1E>,0                ;6B
                    01  0250  2535          .BYTE   ^X01
                        0251  2536          VBUSDATA        <^X1E>,0                ;6C
                    01  0252  2537          .BYTE   ^X01
                        0253  2538          VBUSDATA        <^X1E>,0                ;6D
                    01  0254  2539          .BYTE   ^X01
                        0255  2540          VBUSDATA        <^X1E>,0                ;6E
                    01  0256  2541          .BYTE   ^X01
                        0257  2542          VBUSDATA        <^X1E>,0                ;6F
                    01  0258  2543          .BYTE   ^X01
                        0259  2544          VBUSDATA        <^X1E>,0                ;70
                    01  025A  2545          .BYTE   ^X01
                        025B  2546          VBUSDATA        <^X1E>,0                ;71
                    01  025C  2547          .BYTE   ^X01
                        025D  2548          VBUSDATA        <^X1E>,0                ;72
                    01  025E  2549          .BYTE   ^X01
                        025F  2550          VBUSDATA        <^X1E>,0                ;73
                    01  0260  2551          .BYTE   ^X01
```

```
                    0261  2552          VBUSDATA        <^X1E>,0                ;74
01                  0262  2553          .BYTE   ^X01
                    0263  2554          VBUSDATA        <^X1E>,0                ;75
01                  0264  2555          .BYTE   ^X01
                    0265  2556          VBUSDATA        <^X1E>,0                ;76
01                  0266  2557          .BYTE   ^X01
                    0267  2558          VBUSDATA        <^X1E>,1                ;77
01                  0268  2559          .BYTE   ^X01
                    0269  2560          VBUSDATA        <^X1E>,0                ;78
01                  026A  2561          .BYTE   ^X01
                    026B  2562          VBUSDATA        <^X1E>,0                ;79
01                  026C  2563          .BYTE   ^X01
                    026D  2564          VBUSDATA        <^X1E>,0                ;7A
01                  026E  2565          .BYTE   ^X01
                    026F  2566          VBUSDATA        <^X1E>,0                ;7B
01                  0270  2567          .BYTE   ^X01
                    0271  2568          VBUSDATA        <^X1E>,0                ;7C
01                  0272  2569          .BYTE   ^X01
                    0273  2570          VBUSDATA        <^X1E>,0                ;7D
01                  0274  2571          .BYTE   ^X01
                    0275  2572          VBUSDATA        <^X1E>,0                ;7E
01                  0276  2573          .BYTE   ^X01
                    0277  2574          VBUSDATA        <^X1E>,0                ;7F
01                  0278  2575          .BYTE   ^X01
                    0279  2576          VBUSDATA        <^X1E>,0                ;80
01                  027A  2577          .BYTE   ^X01
                    027B  2578          VBUSDATA        <^X1E>,0                ;81
01                  027C  2579          .BYTE   ^X01
                    027D  2580          VBUSDATA        <^X1E>,0                ;82
01                  027E  2581          .BYTE   ^X01
                    027F  2582          VBUSDATA        <^X1E>,0                ;83
01                  0280  2583          .BYTE   ^X01
                    0281  2584          VBUSDATA        <^X1E>,0                ;84
01                  0282  2585          .BYTE   ^X01
                    0283  2586          VBUSDATA        <^X1E>,0                ;85
01                  0284  2587          .BYTE   ^X01
                    0285  2588          VBUSDATA        <^X1E>,0                ;86
01                  0286  2589          .BYTE   ^X01
                    0287  2590          VBUSDATA        <^X1E>,0                ;87
01                  0288  2591          .BYTE   ^X01
                    0289  2592          VBUSDATA        <^X1E>,0                ;88
01                  028A  2593          .BYTE   ^X01
                    028B  2594          VBUSDATA        <^X1E>,0                ;89
01                  028C  2595          .BYTE   ^X01
                    028D  2596          VBUSDATA        <^X1E>,0                ;8A
01                  028E  2597          .BYTE   ^X01
                    028F  2598          VBUSDATA        <^X1E>,0                ;8B
01                  0290  2599          .BYTE   ^X01
                    0291  2600          VBUSDATA        <^X1E>,0                ;8C
01                  0292  2601          .BYTE   ^X01
                    0293  2602          VBUSDATA        <^X1E>,0                ;8D
01                  0294  2603          .BYTE   ^X01
                    0295  2604          VBUSDATA        <^X1E>,0                ;8E
01                  0296  2605          .BYTE   ^X01
                    0297  2606          VBUSDATA        <^X1E>,0                ;8F
```

```
                01  0298  2607          .BYTE   ^X01
                    0299  2608          VBUSDATA        <^X1E>,0                ;90
                01  029A  2609          .BYTE   ^X01
                    029B  2610          VBUSDATA        <^X1E>,0                ;91
                01  029C  2611          .BYTE   ^X01
                    029D  2612          VBUSDATA        <^X1E>,0                ;92
                01  029E  2613          .BYTE   ^X01
                    029F  2614          VBUSDATA        <^X1E>,0                ;93
                01  02A0  2615          .BYTE   ^X01
                    02A1  2616          VBUSDATA        <^X1E>,0                ;94
                01  02A2  2617          .BYTE   ^X01
                    02A3  2618          VBUSDATA        <^X1E>,0                ;95
                01  02A4  2619          .BYTE   ^X01
                    02A5  2620          VBUSDATA        <^X1E>,0                ;96
                01  02A6  2621          .BYTE   ^X01
                    02A7  2622          VBUSDATA        <^X1E>,0                ;97
                01  02A8  2623          .BYTE   ^X01
                    02A9  2624          VBUSDATA        <^X1E>,0                ;98
                01  02AA  2625          .BYTE   ^X01
                    02AB  2626          VBUSDATA        <^X1E>,0                ;99
                01  02AC  2627          .BYTE   ^X01
                    02AD  2628          VBUSDATA        <^X1E>,0                ;9A
                01  02AE  2629          .BYTE   ^X01
                    02AF  2630          VBUSDATA        <^X1E>,0                ;9B
                01  02B0  2631          .BYTE   ^X01
                    02B1  2632          VBUSDATA        <^X1E>,0                ;9C
                01  02B2  2633          .BYTE   ^X01
                    02B3  2634          VBUSDATA        <^X1E>,0                ;9D
                01  02B4  2635          .BYTE   ^X01
                    02B5  2636          VBUSDATA        <^X1E>,0                ;9E
                01  02B6  2637          .BYTE   ^X01
                    02B7  2638          VBUSDATA        <^X1E>,0                ;9F
                01  02B8  2639          .BYTE   ^X01
                    02B9  2640          VBUSDATA        <^X1E>,0                ;A0
                01  02BA  2641          .BYTE   ^X01
                    02BB  2642          VBUSDATA        <^X1E>,0                ;A1
                01  02BC  2643          .BYTE   ^X01
                    02BD  2644          VBUSDATA        <^X1E>,0                ;A2
                01  02BE  2645          .BYTE   ^X01
                    02BF  2646          VBUSDATA        <^X1E>,0                ;A3
                01  02C0  2647          .BYTE   ^X01
                    02C1  2648          VBUSDATA        <^X1E>,0                ;A4
                01  02C2  2649          .BYTE   ^X01
                    02C3  2650          VBUSDATA        <^X1E>,0                ;A5
                01  02C4  2651          .BYTE   ^X01
                    02C5  2652          VBUSDATA        <^X1E>,0                ;A6
                01  02C6  2653          .BYTE   ^X01
                    02C7  2654          VBUSDATA        <^X1E>,0                ;A7
                01  02C8  2655          .BYTE   ^X01
                    02C9  2656          VBUSDATA        <^X1E>,0                ;A8
                01  02CA  2657          .BYTE   ^X01
                    02CB  2658          VBUSDATA        <^X1E>,0                ;A9
                01  02CC  2659          .BYTE   ^X01
                    02CD  2660          VBUSDATA        <^X1E>,0                ;AA
                01  02CE  2661          .BYTE   ^X01
```

```
                02CF  2662          VBUSDATA        <^X1E>,0                ;AB
          01    02D0  2663          .BYTE   ^X01
                02D1  2664          VBUSDATA        <^X1E>,0                ;AC
          01    02D2  2665          .BYTE   ^X01
                02D3  2666          VBUSDATA        <^X1E>,0                ;AD
          01    02D4  2667          .BYTE   ^X01
                02D5  2668          VBUSDATA        <^X1E>,0                ;AE
          01    02D6  2669          .BYTE   ^X01
                02D7  2670          VBUSDATA        <^X1E>,0                ;AF
          01    02D8  2671          .BYTE   ^X01
                02D9  2672          VBUSDATA        <^X1E>,0                ;B0
          01    02DA  2673          .BYTE   ^X01
                02DB  2674          VBUSDATA        <^X1E>,0                ;B1
          01    02DC  2675          .BYTE   ^X01
                02DD  2676          VBUSDATA        <^X1E>,0                ;B2
          01    02DE  2677          .BYTE   ^X01
                02DF  2678          VBUSDATA        <^X1E>,0                ;B3
          01    02E0  2679          .BYTE   ^X01
                02E1  2680          VBUSDATA        <^X1E>,0                ;B4
          01    02E2  2681          .BYTE   ^X01
                02E3  2682          VBUSDATA        <^X1E>,0                ;B5
          01    02E4  2683          .BYTE   ^X01
                02E5  2684          VBUSDATA        <^X1E>,0                ;B6
          01    02E6  2685          .BYTE   ^X01
                02E7  2686          VBUSDATA        <^X1E>,0                ;B7
          01    02E8  2687          .BYTE   ^X01
                02E9  2688          VBUSDATA        <^X1E>,0                ;B8
          01    02EA  2689          .BYTE   ^X01
                02EB  2690          VBUSDATA        <^X1E>,0                ;B9
          01    02EC  2691          .BYTE   ^X01
                02ED  2692          VBUSDATA        <^X1E>,0                ;BA
          01    02EE  2693          .BYTE   ^X01
                02EF  2694          VBUSDATA        <^X1E>,0                ;BB
          01    02F0  2695          .BYTE   ^X01
                02F1  2696          VBUSDATA        <^X1E>,0                ;BC
          01    02F2  2697          .BYTE   ^X01
                02F3  2698          VBUSDATA        <^X1E>,0                ;BD
          01    02F4  2699          .BYTE   ^X01
                02F5  2700          VBUSDATA        <^X1E>,0                ;BE
          01    02F6  2701          .BYTE   ^X01
                02F7  2702          VBUSDATA        <^X1E>,0                ;BF
          01    02F8  2703          .BYTE   ^X01
                02F9  2704          VBUSDATA        <^X1E>,0                ;C0
          01    02FA  2705          .BYTE   ^X01
                02FB  2706          VBUSDATA        <^X1E>,0                ;C1
          01    02FC  2707          .BYTE   ^X01
                02FD  2708          VBUSDATA        <^X1E>,0                ;C2
          01    02FE  2709          .BYTE   ^X01
                02FF  2710          VBUSDATA        <^X1E>,0                ;C3
          01    0300  2711          .BYTE   ^X01
                0301  2712          VBUSDATA        <^X1E>,0                ;C4
          01    0302  2713          .BYTE   ^X01
                0303  2714          VBUSDATA        <^X1E>,0                ;C5
          01    0304  2715          .BYTE   ^X01
                0305  2716          VBUSDATA        <^X1E>,0                ;C6
```

```
01  0306  2717          .BYTE   ^X01
    0307  2718          VBUSDATA        <^X1E>,0                ;C7
01  0308  2719          .BYTE   ^X01
    0309  2720          VBUSDATA        <^X1E>,0                ;C8
01  030A  2721          .BYTE   ^X01
    030B  2722          VBUSDATA        <^X1E>,0                ;C9
01  030C  2723          .BYTE   ^X01
    030D  2724          VBUSDATA        <^X1E>,0                ;CA
01  030E  2725          .BYTE   ^X01
    030F  2726          VBUSDATA        <^X1E>,0                ;CB
01  0310  2727          .BYTE   ^X01
    0311  2728          VBUSDATA        <^X1E>,0                ;CC
01  0312  2729          .BYTE   ^X01
    0313  2730          VBUSDATA        <^X1E>,0                ;CD
01  0314  2731          .BYTE   ^X01
    0315  2732          VBUSDATA        <^X1E>,0                ;CE
01  0316  2733          .BYTE   ^X01
    0317  2734          VBUSDATA        <^X1E>,0                ;CF
01  0318  2735          .BYTE   ^X01
    0319  2736          VBUSDATA        <^X1E>,0                ;D0
01  031A  2737          .BYTE   ^X01
    031B  2738          VBUSDATA        <^X1E>,0                ;D1
01  031C  2739          .BYTE   ^X01
    031D  2740          VBUSDATA        <^X1E>,0                ;D2
01  031E  2741          .BYTE   ^X01
    031F  2742          VBUSDATA        <^X1E>,0                ;D3
01  0320  2743          .BYTE   ^X01
    0321  2744          VBUSDATA        <^X1E>,0                ;D4
01  0322  2745          .BYTE   ^X01
    0323  2746          VBUSDATA        <^X1E>,0                ;D5
01  0324  2747          .BYTE   ^X01
    0325  2748          VBUSDATA        <^X1E>,0                ;D6
01  0326  2749          .BYTE   ^X01
    0327  2750          VBUSDATA        <^X1E>,0                ;D7
01  0328  2751          .BYTE   ^X01
    0329  2752          VBUSDATA        <^X1E>,0                ;D8
01  032A  2753          .BYTE   ^X01
    032B  2754          VBUSDATA        <^X1E>,0                ;D9
01  032C  2755          .BYTE   ^X01
    032D  2756          VBUSDATA        <^X1E>,0                ;DA
01  032E  2757          .BYTE   ^X01
    032F  2758          VBUSDATA        <^X1E>,0                ;DB
01  0330  2759          .BYTE   ^X01
    0331  2760          VBUSDATA        <^X1E>,0                ;DC
01  0332  2761          .BYTE   ^X01
    0333  2762          VBUSDATA        <^X1E>,0                ;DD
01  0334  2763          .BYTE   ^X01
    0335  2764          VBUSDATA        <^X1E>,0                ;DE
01  0336  2765          .BYTE   ^X01
    0337  2766          VBUSDATA        <^X1E>,0                ;DF
01  0338  2767          .BYTE   ^X01
    0339  2768          VBUSDATA        <^X1E>,0                ;E0
01  033A  2769          .BYTE   ^X01
    033B  2770          VBUSDATA        <^X1E>,0                ;E1
01  033C  2771          .BYTE   ^X01
```

```
                033D  2772          VBUSDATA        <^X1E>,0                ;E2
          01    033E  2773          .BYTE   ^X01
                033F  2774          VBUSDATA        <^X1E>,0                ;E3
          01    0340  2775          .BYTE   ^X01
                0341  2776          VBUSDATA        <^X1E>,0                ;E4
          01    0342  2777          .BYTE   ^X01
                0343  2778          VBUSDATA        <^X1E>,0                ;E5
          01    0344  2779          .BYTE   ^X01
                0345  2780          VBUSDATA        <^X1E>,0                ;E6
          01    0346  2781          .BYTE   ^X01
                0347  2782          VBUSDATA        <^X1E>,0                ;E7
          01    0348  2783          .BYTE   ^X01
                0349  2784          VBUSDATA        <^X1E>,0                ;E8
          01    034A  2785          .BYTE   ^X01
                034B  2786          VBUSDATA        <^X1E>,0                ;E9
          01    034C  2787          .BYTE   ^X01
                034D  2788          VBUSDATA        <^X1E>,0                ;EA
          01    034E  2789          .BYTE   ^X01
                034F  2790          VBUSDATA        <^X1E>,0                ;EB
          01    0350  2791          .BYTE   ^X01
                0351  2792          VBUSDATA        <^X1E>,0                ;EC
          01    0352  2793          .BYTE   ^X01
                0353  2794          VBUSDATA        <^X1E>,0                ;ED
          01    0354  2795          .BYTE   ^X01
                0355  2796          VBUSDATA        <^X1E>,0                ;EE
          01    0356  2797          .BYTE   ^X01
                0357  2798          VBUSDATA        <^X1E>,0                ;EF
          01    0358  2799          .BYTE   ^X01
                0359  2800          VBUSDATA        <^X1E>,0                ;F0
          01    035A  2801          .BYTE   ^X01
                035B  2802          VBUSDATA        <^X1E>,0                ;F1
          01    035C  2803          .BYTE   ^X01
                035D  2804          VBUSDATA        <^X1E>,0                ;F2
          01    035E  2805          .BYTE   ^X01
                035F  2806          VBUSDATA        <^X1E>,0                ;F3
          01    0360  2807          .BYTE   ^X01
                0361  2808          VBUSDATA        <^X1E>,0                ;F4
          01    0362  2809          .BYTE   ^X01
                0363  2810          VBUSDATA        <^X1E>,0                ;F5
          01    0364  2811          .BYTE   ^X01
                0365  2812          VBUSDATA        <^X1E>,0                ;F6
          01    0366  2813          .BYTE   ^X01
                0367  2814          VBUSDATA        <^X1E>,0                ;F7
          01    0368  2815          .BYTE   ^X01
                0369  2816          VBUSDATA        <^X1E>,0                ;F8
          01    036A  2817          .BYTE   ^X01
                036B  2818          VBUSDATA        <^X1E>,0                ;F9
          01    036C  2819          .BYTE   ^X01
                036D  2820          VBUSDATA        <^X1E>,0                ;FA
          01    036E  2821          .BYTE   ^X01
                036F  2822          VBUSDATA        <^X1E>,0                ;FB
          01    0370  2823          .BYTE   ^X01
                0371  2824          VBUSDATA        <^X1E>,1                ;FC
          01    0372  2825          .BYTE   ^X01
                0373  2826          VBUSDATA        <^X1E>,1                ;FD
```

```
01  0374  2827          .BYTE   ^X01
    0375  2828          VBUSDATA        <^X1E>,1                ;FE
01  0376  2829          .BYTE   ^X01
    0377  2830          VBUSDATA        <^X1E>,1                ;FF
    0378  2831 ;
    0378  2832 ; Expected VBUS bit values for Subtest 3
    0378  2833 ;
    0378  2834 ; The following is a table of all the VBUS bit values for an IRD1 with the FPA
    0378  2835 ; enabled. The hex numbers in the comments to the right indicate the range of
    0378  2836 ; loop counter values for each row of address words.
    0378  2837 ;
    0378  2838
01  0378  2839 11$:    .BYTE   ^X01
    0379  2840          VBUSDATA        <^X1E>,1                ;00
01  037A  2841          .BYTE   ^X01
    037B  2842          VBUSDATA        <^X1E>,1                ;01
01  037C  2843          .BYTE   ^X01
    037D  2844          VBUSDATA        <^X1E>,1                ;02
01  037E  2845          .BYTE   ^X01
    037F  2846          VBUSDATA        <^X1E>,1                ;03
01  0380  2847          .BYTE   ^X01
    0381  2848          VBUSDATA        <^X1E>,1                ;04
01  0382  2849          .BYTE   ^X01
    0383  2850          VBUSDATA        <^X1E>,1                ;05
01  0384  2851          .BYTE   ^X01
    0385  2852          VBUSDATA        <^X1E>,1                ;06
01  0386  2853          .BYTE   ^X01
    0387  2854          VBUSDATA        <^X1E>,1                ;07
01  0388  2855          .BYTE   ^X01
    0389  2856          VBUSDATA        <^X1E>,0                ;08
01  038A  2857          .BYTE   ^X01
    038B  2858          VBUSDATA        <^X1E>,0                ;09
01  038C  2859          .BYTE   ^X01
    038D  2860          VBUSDATA        <^X1E>,0                ;0A
01  038E  2861          .BYTE   ^X01
    038F  2862          VBUSDATA        <^X1E>,0                ;0B
01  0390  2863          .BYTE   ^X01
    0391  2864          VBUSDATA        <^X1E>,0                ;0C
01  0392  2865          .BYTE   ^X01
    0393  2866          VBUSDATA        <^X1E>,0                ;0D
01  0394  2867          .BYTE   ^X01
    0395  2868          VBUSDATA        <^X1E>,0                ;0E
01  0396  2869          .BYTE   ^X01
    0397  2870          VBUSDATA        <^X1E>,0                ;0F
01  0398  2871          .BYTE   ^X01
    0399  2872          VBUSDATA        <^X1E>,0                ;10
01  039A  2873          .BYTE   ^X01
    039B  2874          VBUSDATA        <^X1E>,0                ;11
01  039C  2875          .BYTE   ^X01
    039D  2876          VBUSDATA        <^X1E>,0                ;12
01  039E  2877          .BYTE   ^X01
    039F  2878          VBUSDATA        <^X1E>,0                ;13
01  03A0  2879          .BYTE   ^X01
    03A1  2880          VBUSDATA        <^X1E>,0                ;14
01  03A2  2881          .BYTE   ^X01
```

```
                03A3  2882          VBUSDATA        <^X1E>,0                ;15
             01 03A4  2883          .BYTE   ^X01
                03A5  2884          VBUSDATA        <^X1E>,0                ;16
             01 03A6  2885          .BYTE   ^X01
                03A7  2886          VBUSDATA        <^X1E>,0                ;17
             01 03A8  2887          .BYTE   ^X01
                03A9  2888          VBUSDATA        <^X1E>,0                ;18
             01 03AA  2889          .BYTE   ^X01
                03AB  2890          VBUSDATA        <^X1E>,0                ;19
             01 03AC  2891          .BYTE   ^X01
                03AD  2892          VBUSDATA        <^X1E>,0                ;1A
             01 03AE  2893          .BYTE   ^X01
                03AF  2894          VBUSDATA        <^X1E>,0                ;1B
             01 03B0  2895          .BYTE   ^X01
                03B1  2896          VBUSDATA        <^X1E>,0                ;1C
             01 03B2  2897          .BYTE   ^X01
                03B3  2898          VBUSDATA        <^X1E>,0                ;1D
             01 03B4  2899          .BYTE   ^X01
                03B5  2900          VBUSDATA        <^X1E>,0                ;1E
             01 03B6  2901          .BYTE   ^X01
                03B7  2902          VBUSDATA        <^X1E>,0                ;1F
             01 03B8  2903          .BYTE   ^X01
                03B9  2904          VBUSDATA        <^X1E>,0                ;20
             01 03BA  2905          .BYTE   ^X01
                03BB  2906          VBUSDATA        <^X1E>,0                ;21
             01 03BC  2907          .BYTE   ^X01
                03BD  2908          VBUSDATA        <^X1E>,0                ;22
             01 03BE  2909          .BYTE   ^X01
                03BF  2910          VBUSDATA        <^X1E>,0                ;23
             01 03C0  2911          .BYTE   ^X01
                03C1  2912          VBUSDATA        <^X1E>,0                ;24
             01 03C2  2913          .BYTE   ^X01
                03C3  2914          VBUSDATA        <^X1E>,0                ;25
             01 03C4  2915          .BYTE   ^X01
                03C5  2916          VBUSDATA        <^X1E>,0                ;26
             01 03C6  2917          .BYTE   ^X01
                03C7  2918          VBUSDATA        <^X1E>,0                ;27
             01 03C8  2919          .BYTE   ^X01
                03C9  2920          VBUSDATA        <^X1E>,0                ;28
             01 03CA  2921          .BYTE   ^X01
                03CB  2922          VBUSDATA        <^X1E>,0                ;29
             01 03CC  2923          .BYTE   ^X01
                03CD  2924          VBUSDATA        <^X1E>,0                ;2A
             01 03CE  2925          .BYTE   ^X01
                03CF  2926          VBUSDATA        <^X1E>,0                ;2B
             01 03D0  2927          .BYTE   ^X01
                03D1  2928          VBUSDATA        <^X1E>,0                ;2C
             01 03D2  2929          .BYTE   ^X01
                03D3  2930          VBUSDATA        <^X1E>,0                ;2D
             01 03D4  2931          .BYTE   ^X01
                03D5  2932          VBUSDATA        <^X1E>,0                ;2E
             01 03D6  2933          .BYTE   ^X01
                03D7  2934          VBUSDATA        <^X1E>,0                ;2F
             01 03D8  2935          .BYTE   ^X01
                03D9  2936          VBUSDATA        <^X1E>,0                ;30
```

```
                    01  03DA  2937          .BYTE   ^X01
                        03DB  2938          VBUSDATA        <^X1E>,0                ;31
                    01  03DC  2939          .BYTE   ^X01
                        03DD  2940          VBUSDATA        <^X1E>,0                ;32
                    01  03DE  2941          .BYTE   ^X01
                        03DF  2942          VBUSDATA        <^X1E>,0                ;33
                    01  03E0  2943          .BYTE   ^X01
                        03E1  2944          VBUSDATA        <^X1E>,0                ;34
                    01  03E2  2945          .BYTE   ^X01
                        03E3  2946          VBUSDATA        <^X1E>,0                ;35
                    01  03E4  2947          .BYTE   ^X01
                        03E5  2948          VBUSDATA        <^X1E>,0                ;36
                    01  03E6  2949          .BYTE   ^X01
                        03E7  2950          VBUSDATA        <^X1E>,0                ;37
                    01  03E8  2951          .BYTE   ^X01
                        03E9  2952          VBUSDATA        <^X1E>,0                ;38
                    01  03EA  2953          .BYTE   ^X01
                        03EB  2954          VBUSDATA        <^X1E>,0                ;39
                    01  03EC  2955          .BYTE   ^X01
                        03ED  2956          VBUSDATA        <^X1E>,0                ;3A
                    01  03EE  2957          .BYTE   ^X01
                        03EF  2958          VBUSDATA        <^X1E>,0                ;3B
                    01  03F0  2959          .BYTE   ^X01
                        03F1  2960          VBUSDATA        <^X1E>,0                ;3C
                    01  03F2  2961          .BYTE   ^X01
                        03F3  2962          VBUSDATA        <^X1E>,0                ;3D
                    01  03F4  2963          .BYTE   ^X01
                        03F5  2964          VBUSDATA        <^X1E>,0                ;3E
                    01  03F6  2965          .BYTE   ^X01
                        03F7  2966          VBUSDATA        <^X1E>,0                ;3F
                    01  03F8  2967          .BYTE   ^X01
                        03F9  2968          VBUSDATA        <^X1E>,0                ;40
                    01  03FA  2969          .BYTE   ^X01
                        03FB  2970          VBUSDATA        <^X1E>,0                ;41
                    01  03FC  2971          .BYTE   ^X01
                        03FD  2972          VBUSDATA        <^X1E>,0                ;42
                    01  03FE  2973          .BYTE   ^X01
                        03FF  2974          VBUSDATA        <^X1E>,0                ;43
                    01  0400  2975          .BYTE   ^X01
                        0401  2976          VBUSDATA        <^X1E>,0                ;44
                    01  0402  2977          .BYTE   ^X01
                        0403  2978          VBUSDATA        <^X1E>,0                ;45
                    01  0404  2979          .BYTE   ^X01
                        0405  2980          VBUSDATA        <^X1E>,0                ;46
                    01  0406  2981          .BYTE   ^X01
                        0407  2982          VBUSDATA        <^X1E>,0                ;47
                    01  0408  2983          .BYTE   ^X01
                        0409  2984          VBUSDATA        <^X1E>,0                ;48
                    01  040A  2985          .BYTE   ^X01
                        040B  2986          VBUSDATA        <^X1E>,0                ;49
                    01  040C  2987          .BYTE   ^X01
                        040D  2988          VBUSDATA        <^X1E>,0                ;4A
                    01  040E  2989          .BYTE   ^X01
                        040F  2990          VBUSDATA        <^X1E>,0                ;4B
                    01  0410  2991          .BYTE   ^X01
```

```
                0411  2992          VBUSDATA        <^X1E>,0                ;4C
01              0412  2993          .BYTE   ^X01
                0413  2994          VBUSDATA        <^X1E>,0                ;4D
01              0414  2995          .BYTE   ^X01
                0415  2996          VBUSDATA        <^X1E>,0                ;4E
01              0416  2997          .BYTE   ^X01
                0417  2998          VBUSDATA        <^X1E>,0                ;4F
01              0418  2999          .BYTE   ^X01
                0419  3000          VBUSDATA        <^X1E>,0                ;50
01              041A  3001          .BYTE   ^X01
                041B  3002          VBUSDATA        <^X1E>,0                ;51
01              041C  3003          .BYTE   ^X01
                041D  3004          VBUSDATA        <^X1E>,0                ;52
01              041E  3005          .BYTE   ^X01
                041F  3006          VBUSDATA        <^X1E>,0                ;53
01              0420  3007          .BYTE   ^X01
                0421  3008          VBUSDATA        <^X1E>,0                ;54
01              0422  3009          .BYTE   ^X01
                0423  3010          VBUSDATA        <^X1E>,0                ;55
01              0424  3011          .BYTE   ^X01
                0425  3012          VBUSDATA        <^X1E>,0                ;56
01              0426  3013          .BYTE   ^X01
                0427  3014          VBUSDATA        <^X1E>,1                ;57
01              0428  3015          .BYTE   ^X01
                0429  3016          VBUSDATA        <^X1E>,0                ;58
01              042A  3017          .BYTE   ^X01
                042B  3018          VBUSDATA        <^X1E>,1                ;59
01              042C  3019          .BYTE   ^X01
                042D  3020          VBUSDATA        <^X1E>,1                ;5A
01              042E  3021          .BYTE   ^X01
                042F  3022          VBUSDATA        <^X1E>,1                ;5B
01              0430  3023          .BYTE   ^X01
                0431  3024          VBUSDATA        <^X1E>,0                ;5C
01              0432  3025          .BYTE   ^X01
                0433  3026          VBUSDATA        <^X1E>,0                ;5D
01              0434  3027          .BYTE   ^X01
                0435  3028          VBUSDATA        <^X1E>,0                ;5E
01              0436  3029          .BYTE   ^X01
                0437  3030          VBUSDATA        <^X1E>,0                ;5F
01              0438  3031          .BYTE   ^X01
                0439  3032          VBUSDATA        <^X1E>,0                ;60
01              043A  3033          .BYTE   ^X01
                043B  3034          VBUSDATA        <^X1E>,0                ;61
01              043C  3035          .BYTE   ^X01
                043D  3036          VBUSDATA        <^X1E>,0                ;62
01              043E  3037          .BYTE   ^X01
                043F  3038          VBUSDATA        <^X1E>,0                ;63
01              0440  3039          .BYTE   ^X01
                0441  3040          VBUSDATA        <^X1E>,0                ;64
01              0442  3041          .BYTE   ^X01
                0443  3042          VBUSDATA        <^X1E>,0                ;65
01              0444  3043          .BYTE   ^X01
                0445  3044          VBUSDATA        <^X1E>,0                ;66
01              0446  3045          .BYTE   ^X01
                0447  3046          VBUSDATA        <^X1E>,0                ;67
```

```
01  0448  3047          .BYTE   ^X01
    0449  3048          VBUSDATA        <^X1E>,0                ;68
01  044A  3049          .BYTE   ^X01
    044B  3050          VBUSDATA        <^X1E>,0                ;69
01  044C  3051          .BYTE   ^X01
    044D  3052          VBUSDATA        <^X1E>,0                ;6A
01  044E  3053          .BYTE   ^X01
    044F  3054          VBUSDATA        <^X1E>,0                ;6B
01  0450  3055          .BYTE   ^X01
    0451  3056          VBUSDATA        <^X1E>,0                ;6C
01  0452  3057          .BYTE   ^X01
    0453  3058          VBUSDATA        <^X1E>,0                ;6D
01  0454  3059          .BYTE   ^X01
    0455  3060          VBUSDATA        <^X1E>,0                ;6E
01  0456  3061          .BYTE   ^X01
    0457  3062          VBUSDATA        <^X1E>,0                ;6F
01  0458  3063          .BYTE   ^X01
    0459  3064          VBUSDATA        <^X1E>,0                ;70
01  045A  3065          .BYTE   ^X01
    045B  3066          VBUSDATA        <^X1E>,0                ;71
01  045C  3067          .BYTE   ^X01
    045D  3068          VBUSDATA        <^X1E>,0                ;72
01  045E  3069          .BYTE   ^X01
    045F  3070          VBUSDATA        <^X1E>,0                ;73
01  0460  3071          .BYTE   ^X01
    0461  3072          VBUSDATA        <^X1E>,0                ;74
01  0462  3073          .BYTE   ^X01
    0463  3074          VBUSDATA        <^X1E>,0                ;75
01  0464  3075          .BYTE   ^X01
    0465  3076          VBUSDATA        <^X1E>,0                ;76
01  0466  3077          .BYTE   ^X01
    0467  3078          VBUSDATA        <^X1E>,1                ;77
01  0468  3079          .BYTE   ^X01
    0469  3080          VBUSDATA        <^X1E>,0                ;78
01  046A  3081          .BYTE   ^X01
    046B  3082          VBUSDATA        <^X1E>,0                ;79
01  046C  3083          .BYTE   ^X01
    046D  3084          VBUSDATA        <^X1E>,0                ;7A
01  046E  3085          .BYTE   ^X01
    046F  3086          VBUSDATA        <^X1E>,0                ;7B
01  0470  3087          .BYTE   ^X01
    0471  3088          VBUSDATA        <^X1E>,0                ;7C
01  0472  3089          .BYTE   ^X01
    0473  3090          VBUSDATA        <^X1E>,0                ;7D
01  0474  3091          .BYTE   ^X01
    0475  3092          VBUSDATA        <^X1E>,0                ;7E
01  0476  3093          .BYTE   ^X01
    0477  3094          VBUSDATA        <^X1E>,0                ;7F
01  0478  3095          .BYTE   ^X01
    0479  3096          VBUSDATA        <^X1E>,0                ;80
01  047A  3097          .BYTE   ^X01
    047B  3098          VBUSDATA        <^X1E>,0                ;81
01  047C  3099          .BYTE   ^X01
    047D  3100          VBUSDATA        <^X1E>,0                ;82
01  047E  3101          .BYTE   ^X01
```

```
                    047F  3102          VBUSDATA        <^X1E>,0                ;83
                01  0480  3103          .BYTE   ^X01
                    0481  3104          VBUSDATA        <^X1E>,0                ;84
                01  0482  3105          .BYTE   ^X01
                    0483  3106          VBUSDATA        <^X1E>,0                ;85
                01  0484  3107          .BYTE   ^X01
                    0485  3108          VBUSDATA        <^X1E>,0                ;86
                01  0486  3109          .BYTE   ^X01
                    0487  3110          VBUSDATA        <^X1E>,0                ;87
                01  0488  3111          .BYTE   ^X01
                    0489  3112          VBUSDATA        <^X1E>,0                ;88
                01  048A  3113          .BYTE   ^X01
                    048B  3114          VBUSDATA        <^X1E>,0                ;89
                01  048C  3115          .BYTE   ^X01
                    048D  3116          VBUSDATA        <^X1E>,0                ;8A
                01  048E  3117          .BYTE   ^X01
                    048F  3118          VBUSDATA        <^X1E>,0                ;8B
                01  0490  3119          .BYTE   ^X01
                    0491  3120          VBUSDATA        <^X1E>,0                ;8C
                01  0492  3121          .BYTE   ^X01
                    0493  3122          VBUSDATA        <^X1E>,0                ;8D
                01  0494  3123          .BYTE   ^X01
                    0495  3124          VBUSDATA        <^X1E>,0                ;8E
                01  0496  3125          .BYTE   ^X01
                    0497  3126          VBUSDATA        <^X1E>,0                ;8F
                01  0498  3127          .BYTE   ^X01
                    0499  3128          VBUSDATA        <^X1E>,0                ;90
                01  049A  3129          .BYTE   ^X01
                    049B  3130          VBUSDATA        <^X1E>,0                ;91
                01  049C  3131          .BYTE   ^X01
                    049D  3132          VBUSDATA        <^X1E>,0                ;92
                01  049E  3133          .BYTE   ^X01
                    049F  3134          VBUSDATA        <^X1E>,0                ;93
                01  04A0  3135          .BYTE   ^X01
                    04A1  3136          VBUSDATA        <^X1E>,0                ;94
                01  04A2  3137          .BYTE   ^X01
                    04A3  3138          VBUSDATA        <^X1E>,0                ;95
                01  04A4  3139          .BYTE   ^X01
                    04A5  3140          VBUSDATA        <^X1E>,0                ;96
                01  04A6  3141          .BYTE   ^X01
                    04A7  3142          VBUSDATA        <^X1E>,0                ;97
                01  04A8  3143          .BYTE   ^X01
                    04A9  3144          VBUSDATA        <^X1E>,0                ;98
                01  04AA  3145          .BYTE   ^X01
                    04AB  3146          VBUSDATA        <^X1E>,0                ;99
                01  04AC  3147          .BYTE   ^X01
                    04AD  3148          VBUSDATA        <^X1E>,0                ;9A
                01  04AE  3149          .BYTE   ^X01
                    04AF  3150          VBUSDATA        <^X1E>,0                ;9B
                01  04B0  3151          .BYTE   ^X01
                    04B1  3152          VBUSDATA        <^X1E>,0                ;9C
                01  04B2  3153          .BYTE   ^X01
                    04B3  3154          VBUSDATA        <^X1E>,0                ;9D
                01  04B4  3155          .BYTE   ^X01
                    04B5  3156          VBUSDATA        <^X1E>,0                ;9E
```

```
                    01  04B6  3157          .BYTE   ^X01
                        04B7  3158          VBUSDATA        <^X1E>,0                ;9F
                    01  04B8  3159          .BYTE   ^X01
                        04B9  3160          VBUSDATA        <^X1E>,0                ;A0
                    01  04BA  3161          .BYTE   ^X01
                        04BB  3162          VBUSDATA        <^X1E>,0                ;A1
                    01  04BC  3163          .BYTE   ^X01
                        04BD  3164          VBUSDATA        <^X1E>,0                ;A2
                    01  04BE  3165          .BYTE   ^X01
                        04BF  3166          VBUSDATA        <^X1E>,0                ;A3
                    01  04C0  3167          .BYTE   ^X01
                        04C1  3168          VBUSDATA        <^X1E>,0                ;A4
                    01  04C2  3169          .BYTE   ^X01
                        04C3  3170          VBUSDATA        <^X1E>,0                ;A5
                    01  04C4  3171          .BYTE   ^X01
                        04C5  3172          VBUSDATA        <^X1E>,0                ;A6
                    01  04C6  3173          .BYTE   ^X01
                        04C7  3174          VBUSDATA        <^X1E>,0                ;A7
                    01  04C8  3175          .BYTE   ^X01
                        04C9  3176          VBUSDATA        <^X1E>,0                ;A8
                    01  04CA  3177          .BYTE   ^X01
                        04CB  3178          VBUSDATA        <^X1E>,0                ;A9
                    01  04CC  3179          .BYTE   ^X01
                        04CD  3180          VBUSDATA        <^X1E>,0                ;AA
                    01  04CE  3181          .BYTE   ^X01
                        04CF  3182          VBUSDATA        <^X1E>,0                ;AB
                    01  04D0  3183          .BYTE   ^X01
                        04D1  3184          VBUSDATA        <^X1E>,0                ;AC
                    01  04D2  3185          .BYTE   ^X01
                        04D3  3186          VBUSDATA        <^X1E>,0                ;AD
                    01  04D4  3187          .BYTE   ^X01
                        04D5  3188          VBUSDATA        <^X1E>,0                ;AE
                    01  04D6  3189          .BYTE   ^X01
                        04D7  3190          VBUSDATA        <^X1E>,0                ;AF
                    01  04D8  3191          .BYTE   ^X01
                        04D9  3192          VBUSDATA        <^X1E>,0                ;B0
                    01  04DA  3193          .BYTE   ^X01
                        04DB  3194          VBUSDATA        <^X1E>,0                ;B1
                    01  04DC  3195          .BYTE   ^X01
                        04DD  3196          VBUSDATA        <^X1E>,0                ;B2
                    01  04DE  3197          .BYTE   ^X01
                        04DF  3198          VBUSDATA        <^X1E>,0                ;B3
                    01  04E0  3199          .BYTE   ^X01
                        04E1  3200          VBUSDATA        <^X1E>,0                ;B4
                    01  04E2  3201          .BYTE   ^X01
                        04E3  3202          VBUSDATA        <^X1E>,0                ;B5
                    01  04E4  3203          .BYTE   ^X01
                        04E5  3204          VBUSDATA        <^X1E>,0                ;B6
                    01  04E6  3205          .BYTE   ^X01
                        04E7  3206          VBUSDATA        <^X1E>,0                ;B7
                    01  04E8  3207          .BYTE   ^X01
                        04E9  3208          VBUSDATA        <^X1E>,0                ;B8
                    01  04EA  3209          .BYTE   ^X01
                        04EB  3210          VBUSDATA        <^X1E>,0                ;B9
                    01  04EC  3211          .BYTE   ^X01
```

```
        04ED  3212        VBUSDATA        <^X1E>,0                ;BA
   01   04EE  3213        .BYTE   ^X01
        04EF  3214        VBUSDATA        <^X1E>,0                ;BB
   01   04F0  3215        .BYTE   ^X01
        04F1  3216        VBUSDATA        <^X1E>,0                ;BC
   01   04F2  3217        .BYTE   ^X01
        04F3  3218        VBUSDATA        <^X1E>,0                ;BD
   01   04F4  3219        .BYTE   ^X01
        04F5  3220        VBUSDATA        <^X1E>,0                ;BE
   01   04F6  3221        .BYTE   ^X01
        04F7  3222        VBUSDATA        <^X1E>,0                ;BF
   01   04F8  3223        .BYTE   ^X01
        04F9  3224        VBUSDATA        <^X1E>,0                ;C0
   01   04FA  3225        .BYTE   ^X01
        04FB  3226        VBUSDATA        <^X1E>,0                ;C1
   01   04FC  3227        .BYTE   ^X01
        04FD  3228        VBUSDATA        <^X1E>,0                ;C2
   01   04FE  3229        .BYTE   ^X01
        04FF  3230        VBUSDATA        <^X1E>,0                ;C3
   01   0500  3231        .BYTE   ^X01
        0501  3232        VBUSDATA        <^X1E>,0                ;C4
   01   0502  3233        .BYTE   ^X01
        0503  3234        VBUSDATA        <^X1E>,0                ;C5
   01   0504  3235        .BYTE   ^X01
        0505  3236        VBUSDATA        <^X1E>,0                ;C6
   01   0506  3237        .BYTE   ^X01
        0507  3238        VBUSDATA        <^X1E>,0                ;C7
   01   0508  3239        .BYTE   ^X01
        0509  3240        VBUSDATA        <^X1E>,0                ;C8
   01   050A  3241        .BYTE   ^X01
        050B  3242        VBUSDATA        <^X1E>,0                ;C9
   01   050C  3243        .BYTE   ^X01
        050D  3244        VBUSDATA        <^X1E>,0                ;CA
   01   050E  3245        .BYTE   ^X01
        050F  3246        VBUSDATA        <^X1E>,0                ;CB
   01   0510  3247        .BYTE   ^X01
        0511  3248        VBUSDATA        <^X1E>,0                ;CC
   01   0512  3249        .BYTE   ^X01
        0513  3250        VBUSDATA        <^X1E>,0                ;CD
   01   0514  3251        .BYTE   ^X01
        0515  3252        VBUSDATA        <^X1E>,0                ;CE
   01   0516  3253        .BYTE   ^X01
        0517  3254        VBUSDATA        <^X1E>,0                ;CF
   01   0518  3255        .BYTE   ^X01
        0519  3256        VBUSDATA        <^X1E>,0                ;D0
   01   051A  3257        .BYTE   ^X01
        051B  3258        VBUSDATA        <^X1E>,0                ;D1
   01   051C  3259        .BYTE   ^X01
        051D  3260        VBUSDATA        <^X1E>,0                ;D2
   01   051E  3261        .BYTE   ^X01
        051F  3262        VBUSDATA        <^X1E>,0                ;D3
   01   0520  3263        .BYTE   ^X01
        0521  3264        VBUSDATA        <^X1E>,0                ;D4
   01   0522  3265        .BYTE   ^X01
        0523  3266        VBUSDATA        <^X1E>,0                ;D5
```

```
                01  0524  3267          .BYTE   ^X01
                    0525  3268          VBUSDATA        <^X1E>,0                ;D6
                01  0526  3269          .BYTE   ^X01
                    0527  3270          VBUSDATA        <^X1E>,0                ;D7
                01  0528  3271          .BYTE   ^X01
                    0529  3272          VBUSDATA        <^X1E>,0                ;D8
                01  052A  3273          .BYTE   ^X01
                    052B  3274          VBUSDATA        <^X1E>,0                ;D9
                01  052C  3275          .BYTE   ^X01
                    052D  3276          VBUSDATA        <^X1E>,0                ;DA
                01  052E  3277          .BYTE   ^X01
                    052F  3278          VBUSDATA        <^X1E>,0                ;DB
                01  0530  3279          .BYTE   ^X01
                    0531  3280          VBUSDATA        <^X1E>,0                ;DC
                01  0532  3281          .BYTE   ^X01
                    0533  3282          VBUSDATA        <^X1E>,0                ;DD
                01  0534  3283          .BYTE   ^X01
                    0535  3284          VBUSDATA        <^X1E>,0                ;DE
                01  0536  3285          .BYTE   ^X01
                    0537  3286          VBUSDATA        <^X1E>,0                ;DF
                01  0538  3287          .BYTE   ^X01
                    0539  3288          VBUSDATA        <^X1E>,0                ;E0
                01  053A  3289          .BYTE   ^X01
                    053B  3290          VBUSDATA        <^X1E>,0                ;E1
                01  053C  3291          .BYTE   ^X01
                    053D  3292          VBUSDATA        <^X1E>,0                ;E2
                01  053E  3293          .BYTE   ^X01
                    053F  3294          VBUSDATA        <^X1E>,0                ;E3
                01  0540  3295          .BYTE   ^X01
                    0541  3296          VBUSDATA        <^X1E>,0                ;E4
                01  0542  3297          .BYTE   ^X01
                    0543  3298          VBUSDATA        <^X1E>,0                ;E5
                01  0544  3299          .BYTE   ^X01
                    0545  3300          VBUSDATA        <^X1E>,0                ;E6
                01  0546  3301          .BYTE   ^X01
                    0547  3302          VBUSDATA        <^X1E>,0                ;E7
                01  0548  3303          .BYTE   ^X01
                    0549  3304          VBUSDATA        <^X1E>,0                ;E8
                01  054A  3305          .BYTE   ^X01
                    054B  3306          VBUSDATA        <^X1E>,0                ;E9
                01  054C  3307          .BYTE   ^X01
                    054D  3308          VBUSDATA        <^X1E>,0                ;EA
                01  054E  3309          .BYTE   ^X01
                    054F  3310          VBUSDATA        <^X1E>,0                ;EB
                01  0550  3311          .BYTE   ^X01
                    0551  3312          VBUSDATA        <^X1E>,0                ;EC
                01  0552  3313          .BYTE   ^X01
                    0553  3314          VBUSDATA        <^X1E>,0                ;ED
                01  0554  3315          .BYTE   ^X01
                    0555  3316          VBUSDATA        <^X1E>,0                ;EE
                01  0556  3317          .BYTE   ^X01
                    0557  3318          VBUSDATA        <^X1E>,0                ;EF
                01  0558  3319          .BYTE   ^X01
                    0559  3320          VBUSDATA        <^X1E>,0                ;F0
                01  055A  3321          .BYTE   ^X01
```

```
                        055B  3322          VBUSDATA        <^X1E>,0                ;F1
                    01  055C  3323          .BYTE   ^X01
                        055D  3324          VBUSDATA        <^X1E>,0                ;F2
                    01  055E  3325          .BYTE   ^X01
                        055F  3326          VBUSDATA        <^X1E>,0                ;F3
                    01  0560  3327          .BYTE   ^X01
                        0561  3328          VBUSDATA        <^X1E>,0                ;F4
                    01  0562  3329          .BYTE   ^X01
                        0563  3330          VBUSDATA        <^X1E>,0                ;F5
                    01  0564  3331          .BYTE   ^X01
                        0565  3332          VBUSDATA        <^X1E>,0                ;F6
                    01  0566  3333          .BYTE   ^X01
                        0567  3334          VBUSDATA        <^X1E>,0                ;F7
                    01  0568  3335          .BYTE   ^X01
                        0569  3336          VBUSDATA        <^X1E>,0                ;F8
                    01  056A  3337          .BYTE   ^X01
                        056B  3338          VBUSDATA        <^X1E>,0                ;F9
                    01  056C  3339          .BYTE   ^X01
                        056D  3340          VBUSDATA        <^X1E>,0                ;FA
                    01  056E  3341          .BYTE   ^X01
                        056F  3342          VBUSDATA        <^X1E>,0                ;FB
                    01  0570  3343          .BYTE   ^X01
                        0571  3344          VBUSDATA        <^X1E>,1                ;FC
                    01  0572  3345          .BYTE   ^X01
                        0573  3346          VBUSDATA        <^X1E>,1                ;FD
                    01  0574  3347          .BYTE   ^X01
                        0575  3348          VBUSDATA        <^X1E>,1                ;FE
                    01  0576  3349          .BYTE   ^X01
                        0577  3350          VBUSDATA        <^X1E>,1                ;FF
                        0578  3351
                        0578  3352 BEGIN_CPU_DATA
                        0002  3353
                        0002  3354 DCS_DATA
                        0001  3355
U 00, 7700,C800,0364,0300,0470,1801     0001  3356 ;% 00   NOP                             ; GET UP TO SPEED
U 01, 5700,C840,0364,0304,0470,1802     000C  3360 ;% 01   R[TEMP1]_RDM                   ; LOAD THE OPCODE INTO RTEMP1
U 02, 7700,F800,57D7,D7FF,8470,1803     0017  3364 ;% 02   LONLIT_[50505000]             ; LOAD THE OPERANDS INTO THE LONLIT REG
U 03, 7700,8800,5BE5,03D4,0470,1804     0022  3368 ;% 03   Q_R[LONLIT]                   ; MOVE THE OPERANDS TO THE Q REGISTER
U 04, 7700,0841,00A4,0304,0470,1805     002D  3372 ;% 04   R[TEMP1]_M[TEMP1].OR.Q        ; OR THE OPCODE WITH THE OPERANDS
U 05, 7700,4800,5BE4,03D8,4A70,1806     0038  3376 ;% 05   VA_R[ZERO]                    ; ZERO THE VA
U 06, 7701,C800,5BE4,0204,5590,1807     0043  3380 ;% 06   WRITE.LONG R[TEMP1]          ; WRITE THE MODEL INSTRUCTION TO CACHE
U 07, 7700,8800,5BE4,0304,5470,1808     004E  3384 ;% 07   WDR_UNROT(R[TEMP1])         ; CMI OFF SO HOLD DATA FOR SECOND CYCLE
U 08, 7700,C800,0364,0300,4470,1809     0059  3388 ;% 08   VA_VA+4                      ; INCREMENT THE VA
U 09, 7701,0800,5BE4,02D8,5590,180A     0064  3392 ;% 09   WRITE.LONG R[ZERO]          ; VALIDATE NEXT LOCATION FOR XB1
U 0A, 7700,C800,5BE4,03D8,5470,180B     006F  3396 ;% 0A   WDR_UNROT(R[ZERO])         ; CMI OFF SO HOLD DATA FOR SECOND CYCLE
U 0B, 7700,C800,5BE4,03D8,4870,180C     007A  3400 ;% 0B   PC_R[ZERO]                   ; START INSTRUCTION PREFETCH, LOAD XB'S
U 0C, 7701,0857,5924,0202,0470,180D     0085  3404 ;% 0C   R[TEMP0]_M[XB.PC_PC+I],ISIZE[LONG] ; LOAD RTEMP0 WITH XB0
U 0D, 7700,4800,5BE4,03D8,4870,180E     0090  3408 ;% 0D   PC_R[ZERO]                   ; RESET THE PC AND RELOAD THE XB'S
U 0E, 7600,0800,0364,1300,0470,180F     009B  3412 ;% 0E   BUT/IRD1,STROBE.V.BUS        ; EXECUTE IRD1 AND STROBE VBUS
U 0F, 7300,C800,0364,0300,0470,1810     00A6  3416 ;% 0F   STOP.CLOCK                   ; HALT
                                        00B1  3420
U 10, 7700,4800,0364,0300,0470,1811     00B1  3421 ;% 10   NOP                           ; GET CPU UP TO SPEED
U 11, 5700,4800,0364,0300,2A70,1812     00BC  3425 ;% 11   WCTRL/FPA.ENABLE_WB5,WB_RDM   ; WRITE THE FPA'S ENABLE BIT
U 12, 7700,C800,0364,0300,0470,1813     00C7  3429 ;% 12   NOP                           ; NOP TO GET CPU UP TO SPEED
U 13, 7700,4800,0364,C300,0470,1800     00D2  3433 ;% 13   BUT/NO.FPA,NEXT/1800         ; CHECK FPA STATUS
```

```
U 14. 7300.C800.0364.0300.0470.1815  00DD  3437 ;% 14   STOP.CLOCK                          ; HALT
                                     00E8  3441
U 15. 7700.C800.0364.0300.0470.1816  00E8  3442 ;% 15   NOP                                ; NOP TO GET CPU UP TO SPEED
U 16. 7700.F800.7F7F.7F7F.0470.1817  00F3  3446 ;% 16   LONLIT_[01010101]                  ; LOAD LONGLIT WITH 4 NOP INSTRUCTIONS
U 17. 7700.4800.5BE4.03D8.4A70.1818  00FE  3450 ;% 17   VA_R[ZERO]                        ; ZERO THE VA
U 18. 7701.8800.5BE4.02D4.5590.1819  0109  3454 ;% 18   WRITE.LONG R[LONLIT]              ; WRITE NOPS TO CACHE FOR XB0
U 19. 7700.C800.5BE4.03D4.5470.181A  0114  3458 ;% 19   WDR_UNROT(R[LONLIT])             ; CMI OFF SO HOLD FOR SECOND CYCLE
U 1A. 7700.C800.0364.0300.4470.181B  011F  3462 ;% 1A   VA_VA+4                          ; INCREMENT THE VA
U 1B. 7701.8800.5BE4.02D4.5590.181C  012A  3466 ;% 1B   WRITE.LONG R[LONLIT]              ; VALIDATE NEXT LOCATION FOR XB1
U 1C. 7700.4800.5BE4.03D4.5470.181D  0135  3470 ;% 1C   WDR_UNROT(R[LONLIT])             ; CMI OFF SO HOLD FOR SECOND CYCLE
U 1D. 7700.C800.5BE4.03D8.4870.181E  0140  3474 ;% 1D   PC_R[ZERO]                       ; ZERO THE PC TO LOAD XB'S
U 1E. 7700.C800.0364.1700.0470.181F  014B  3478 ;% 1E   IRD1TEST                        ; IRD1 TO SET FPA FOR NON-FPA OPCODE
U 1F. 7300.C800.0364.0300.0470.1820  0156  3482 ;% 1F   STOP.CLOCK                       ; HALT
                                     0161  3486
                                     0161  3487 RTEMP_DATA
                                     0001  3488
                         00000000    0001  3489         .LONG   0       ; RTEMP0 HOLDS THE CONTENTS OF XB0
                         00000000    0005  3490         .LONG   0       ; RTEMP1 HOLDS THE INSTRUCTION OPCODE
                                     0009  3491
                                     0009  3492 END_CPU_DATA
                                     0578  3493
                                     0578  3494 END_TEST_DATA
                                     0578
                                     0578        ;-------------------------------------------------------------------------
                                     0578  3495
                                     0578  3496 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
                        002B        .SBTTL  "TEST   0C4     IRD1 Instruction Address Decode Test 3
                        002B  3498 ;*******************************************************************************
                        002B  3499 ;++
                        002B  3500 ;
                        002B  3501 ; FUNCTIONAL DESCRIPTION
                        002B  3502 ;
                        002B  3503 ;       This test will examine the ROMS that implement a native mode IRD1 when
                        002B  3504 ;       the First Part Done (FPD) bit of the PSL is set to a one.
                        002B  3505 ;
                        002B  3506 ; TEST ALGORITHM
                        002B  3507 ;
                        002B  3508 ;       This test is implemented in three subtests. The first subtest examines
                        002B  3509 ;       the IRD1 ROMS when the CPU's Floating Point Accelerator (FPA) is not
                        002B  3510 ;       present. The second subtest examines the ROMS when there is an FPA
                        002B  3511 ;       present and it has been disabled. The third subtest examines the ROMS
                        002B  3512 ;       when the FPA is present and enabled. The test algorithm executes as
                        002B  3513 ;       follows:
                        002B  3514 ;
                        002B  3515 ;        1. Initialize the CPU. This will clear the FPD bit in the PSL.
                        002B  3516 ;        2. Execute the following microcode to set the FPD bit in the PSL and
                        002B  3517 ;           to prepare an FPA to be enabled or disabled:
                        002B  3518 ;                1. NOP to get the CPU up to speed.
                        002B  3519 ;                2. Load the data needed to set the FPD bit into longlit.
                        002B  3520 ;                3. Set the FPD bit in the PSL.
                        002B  3521 ;                4. Load four NOP opcodes into the longlit register for XB0.
                        002B  3522 ;                5. Zero the VA.
                        002B  3523 ;                6. Write the NOPS for XB0 into cache at address 0.
                        002B  3524 ;                7. Increment the VA by 4.
                        002B  3525 ;                8. Validate the next location for XB1 prefetch by writing
                        002B  3526 ;                   NOPS to cache at address 4.
                        002B  3527 ;                9. Zero the PC to initiate the instruction prefetch that
                        002B  3528 ;                   writes XB0 and XB1 from cache.
                        002B  3529 ;               10. Execute an IRD1TEST cycle to decode the NOP in XB0. This
                        002B  3530 ;                   will intialize the FPA to allow it to be enabled/disabled.
                        002B  3531 ;               11. STOP.CLOCK to halt.
                        002B  3532 ;        3. Execute the following microcode to test for an FPA presence:
                        002B  3533 ;                12. NOP to get the CPU up to speed.
                        002B  3534 ;                13. BUT/NO.FPA,NEXT/1800
                        002B  3535 ;                14. STOP.CLOCK to halt.
                        002B  3536 ;        4. Perform a masked comparison of the address in the Control Store
                        002B  3537 ;           Address Trace Register (CSTRP) against the expected address for the
                        002B  3538 ;           FPA not present.
                        002B  3539 ;        5. If the two do not match, then there is an FPA, so branch to
                        002B  3540 ;           Subtest 2, otherwise continue on to Subtest 1.
                        002B  3541 ;
                        002B  3542 ;       Subtest 1
                        002B  3543 ;
                        002B  3544 ;        6. Loop on error from here.
                        002B  3545 ;        7. Re-execute the microcode in step 3 to test the FPA enable bit.
                        002B  3546 ;        8. Perform a masked comparison of the CSTRP with the expected address
                        002B  3547 ;           for the FPA disabled.
                        002B  3548 ;        9. If the two do not match then there is a problem with the FPA/CPU
                        002B  3549 ;           interface so signal an error.
                        002B  3550 ;       10. Prepare the RDM to expect non-DCS addresses.
                        002B  3551 ;       11. Prepare a loop on I for 256 VAX opcodes.
```

```
002B  3552 ;       12. Save the loop counter minus one in the RDM memory.
002B  3553 ;       13. Load the WDREG with the number saved in RDM memory.
002B  3554 ;       14. Loop on error from here.
002B  3555 ;       15. Execute the following test microcode:
002B  3556 ;                15. NOP to get the CPU up to speed.
002B  3557 ;                16. Load the opcode to be tested from the WDREG to RTEMP1.
002B  3558 ;                17. Load the operands into the longlit register.
002B  3559 ;                18. Move the operands from the longlit register into the
002B  3560 ;                    Q register.
002B  3561 ;                19. Form the test instruction by ORing the opcode in RTEMP1
002B  3562 ;                    (MTEMP1=RTEMP1) with the operands in the Q register and
002B  3563 ;                    write the result back to RTEMP1.
002B  3564 ;                20. Zero the VA.
002B  3565 ;                21. Write the test instruction in RTEMP1 to cache at address 0.
002B  3566 ;                22. Increment the VA by 4.
002B  3567 ;                23. Validate the next location for XB1 prefetch by writing
002B  3568 ;                    zeros to cache at address 4.
002B  3569 ;                24. Zero the PC to initiate the instruction prefetch that
002B  3570 ;                    writes XB0 and XB1 from cache.
002B  3571 ;                25. Write the contents of XB0 to RTEMP0. This will cause the
002B  3572 ;                    PC to increment by 4 and can change the contents of XB0.
002B  3573 ;                26. Rezero the PC to reload XB0 and XB1.
002B  3574 ;                27. Execute a BUT/IRD1 to decode the instruction in XB0.
002B  3575 ;                28. Halt.
002B  3576 ;       16. Perform a masked comparison of the address in the CSTRP register
002B  3577 ;           with the expected control store address for the IRD1.
002B  3578 ;       17. If the two are not equal, signal an error.
002B  3579 ;       18. Continue to loop on I for the remaining VAX opcodes.
002B  3580 ;       19. Skip to the next test.
002B  3581 ;
002B  3582 ;       Subtests 2 and 3
002B  3583 ;
002B  3584 ;       21. Load the WDREG with a longword to perform an FPA disable for
002B  3585 ;           Subtest 2 or an FPA enable for Subtest 3.
002B  3586 ;       21. Loop on error from here.
002B  3587 ;       22. Execute the following microcode to disable the FPA for Subtest 2 or
002B  3588 ;           to enable the FPA for Subtest 3 and to also check the result:
002B  3589 ;                29. NOP to get the CPU up to speed.
002B  3590 ;                30. WCTRL/FPAENABLE_WB5,WB_RDM to write the FPA's enable bit
002B  3591 ;                    with bit 5 of the RDM's WDREG.
002B  3592 ;                31. NOP
002B  3593 ;                32. BUT/NO.FPA,NEXT/1800 to test for FPA enabled.
002B  3594 ;                33. STOP.CLOCK to halt.
002B  3595 ;       23. Perform a masked comparison of the CSTRP with the expected address
002B  3596 ;           for the FPA disabled (Subtest 2) or the FPA enabled (Subtest 3).
002B  3597 ;       24. If the two do not match then there is a problem with the FPA/CPU
002B  3598 ;           interface so signal an error.
002B  3599 ;       25. Prepare the RDM to expect non-DCS addresses.
002B  3600 ;       26. Prepare a loop on I for 256 VAX opcodes.
002B  3601 ;       27. Save the loop counter minus one in the RDM memory.
002B  3602 ;       28. Load the WDREG with the number saved in RDM memory.
002B  3603 ;       29. Loop on error from here.
002B  3604 ;       30. Execute the same test microcode as in Subtest 1 step 15.
002B  3605 ;       31. Perform a masked comparison of the address in the CSTRP register
002B  3606 ;           with the expected control store address for the IRD1.
```

```
                        002B  3607 ;       32. If the two are not equal, signal an error.
                        002B  3608 ;       33. Continue to loop on I for the remaining VAX opcodes.
                        002B  3609 ;       34. If this is Subtest 2 then continue on to Subtest 3.
                        002B  3610 ;           If this is Subtest 3 then skip to the next test.
                        002B  3611 ;
                        002B  3612 ; TEST PATTERNS
                        002B  3613 ;
                        002B  3614 ;       For this test the VAX machine opcodes are generated by a simple count
                        002B  3615 ;       pattern. The opcodes are based on the 256 base machine instructions
                        002B  3616 ;       as specified in the VAX architecture handbook. The generated opcodes
                        002B  3617 ;       are written to the CPU and combined with three bytes of operand
                        002B  3618 ;       data to form a VAX machine instruction. The operands all specify R0 in
                        002B  3619 ;       register mode and are loaded from the CPU's long literal register.
                        002B  3620 ;
                        002B  3621 ;       This test also requires two tables of expected control store addresses
                        002B  3622 ;       for each IRD1 Instruction Decode Cycle. One table contains the expected
                        002B  3623 ;       addresses for an IRD1 when the FPA has been disabled, the other for
                        002B  3624 ;       when it is enabled.  The expected control store addresses were obtained
                        002B  3625 ;       from the Native Mode Ird Rom Definition in the CPU microcode.
                        002B  3626 ;
                        002B  3627 ; LOGIC DESCRIPTION
                        002B  3628 ;
                        002B  3629 ;       The CCS ROMS are fed addresses from ROMS on the DPM during instruction
                        002B  3630 ;       decode. The DPM ROMS are addressed by bits <07:00> of the execution
                        002B  3631 ;       buffer. These bits form the opcode for the instruction. For zero and
                        002B  3632 ;       certain single operand instructions the addresses generated by the ROMS
                        002B  3633 ;       will force a jump to execution flows in the microcode. For the rest,
                        002B  3634 ;       operand specifier evaulation addresses will be generated.
                        002B  3635 ;
                        002B  3636 ; ASSUMPTIONS
                        002B  3637 ;
                        002B  3638 ;       This test assumes the following logic has already been tested; the
                        002B  3639 ;       CPU's data path, the cache, the XB, and the IR and OSR.
                        002B  3640 ;
                        002B  3641 ; ERROR DESCRIPTION
                        002B  3642 ;
                        002B  3643 ;       On error this test can type out the following information:
                        002B  3644 ;
                        002B  3645 ;       Failing gate arrays:    IRD
                        002B  3646 ;       Failing modules:        FPA, MIC
                        002B  3647 ;
                        002B  3648 ;       The following data will also by typed:
                        002B  3649 ;
                        002B  3650 ;       EXPECTED CONTROL STORE ADDRESS
                        002B  3651 ;       RECEIVED CONTROL STORE ADDRESS
                        002B  3652 ;       LOOP COUNT (INDEX I)
                        002B  3653 ;       THE INSTRUCTION BEING DECODED FROM XB0 (RTEMP0)
                        002B  3654 ;
                        002B  3655 ;--
                        002B  3656 ;*****************************************************************************
                        002B  3657
                        002B  3658         INITIALIZE                                  ; INITIALIZE THE CPU
                        002D  3659         FETCH           <^X15>                      ; SET FPD BIT IN PSL AND PREP.
                        0032  3660         BRSTCLK         <^X21>                      ; AN FPA TO BE ENABLED/DISABLED
                        0036  3661         FETCH           <^X12>                      ; EXECUTE A BUT/NO.FPA TO TEST
```

```
                003B  3662          BRSTCLK         <^X14>                  ; FOR AN FPA PRESENT
                003F  3663          CMPREGMSK       CSTRP,1$,,2$,,W         ; CHECK THE BUT/NO.FPA ADDRESS
                004B  3664          SKIP            100$,ONERROR            ; SKIP IF THE FPA IS PRESENT
                0052  3665
                0052  3666          SUBTEST         1
                0055
                0055        ;////////////////////////////////////////////////////////////////////////////////
                0055  3667  ;+
                0055  3668  ; Test the IRD1 ROMS when the FPA is not present.
                0055  3669  ;-
                0055  3670          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                0057  3671          FETCH           <^X12>                  ; EXECUTE A BUT/NO.FPA TO TEST
                005C  3672          BRSTCLK         <^X14>,INH              ; FOR THE FPA DISABLED
                0060  3673          CMPREGMSK       CSTRP,3$,,5$,,W         ; IS THE FPA DISABLED?
                006C  3674          IFERROR         ,,<FPA>                 ; IF NOT, SIGNAL AN ERROR
                0072  3675          EXPUTRAP                                ; EXPECT NON-DCS ADDRESSES
                0074  3676          LOOP            I,1,256                 ; LOOP FOR 256 OPCODES
                007B  3677          SAVEINDEX       I,9$                    ; SAVE I-1 TO FORM AN OPCODE
                0080  3678          LOADREG         WDREG,9$,,L             ; LOAD WDREG WITH THE OPCODE
                0088  3679          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                008A  3680          FETCH           0                       ; EXECUTE THE IRD1 ADDRESS
                008F  3681          BRSTCLK         <^X0F>,INH              ; TEST MICROCODE
                0093  3682          CMPREGMSK       CSTRP,10$,I,6$,,W       ; IS THE CS ADDRESS CORRECT?
                009F  3683          IFERROR         1,<IRD>,<MIC>           ; IF NOT, SIGNAL AN ERROR
                00A6  3684          ENDLOOP         I                       ; CONTINUE THE LOOP
                00A9  3685          SKIP                                    ; SKIP TO THE NEXT TEST
                00B0  3686
                00B0  3687  100$:   SUBTEST         2
                00B3
                00B3        ;////////////////////////////////////////////////////////////////////////////////
                00B3  3688  ;+
                00B3  3689  ; Test the IRD1 ROMS with the FPA present but not enabled.
                00B3  3690  ;-
                00B3  3691          LOADREG         WDREG,7$,,L             ; LOAD DATA TO DISABLE THE FPA
                00BB  3692          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                00BD  3693          FETCH           <^X10>                  ; EXECUTE AN FPA DISABLE
                00C2  3694          BRSTCLK         <^X14>,INH              ; AND READ BACK THE FPA STATUS
                00C6  3695          CMPREGMSK       CSTRP,3$,,5$,,W         ; IS THE FPA DISABLED?
                00D2  3696          IFERROR         ,,<FPA>                 ; IF NOT, SIGNAL AN ERROR
                00D8  3697          EXPUTRAP                                ; EXPECT NON-DCS ADDRESSES
                00DA  3698          LOOP            I,1,256                 ; LOOP FOR 256 OPCODES
                00E1  3699          SAVEINDEX       I,9$                    ; SAVE I-1 TO FORM AN OPCODE
                00E6  3700          LOADREG         WDREG,9$,,L             ; LOAD WDREG WITH THE OPCODE
                00EE  3701          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                00F0  3702          FETCH           0                       ; EXECUTE THE IRD1 ADDRESS
                00F5  3703          BRSTCLK         <^X0F>,INH              ; TEST MICROCODE
                00F9  3704          CMPREGMSK       CSTRP,10$,I,6$,,W       ; IS THE CS ADDRESS CORRECT?
                0105  3705          IFERROR         1,<IRD>,<MIC>           ; IF NOT, SIGNAL AN ERROR
                010C  3706          ENDLOOP         I                       ; CONTINUE THE LOOP
                010F  3707
                010F  3708          SUBTEST         3
                0112
                0112        ;////////////////////////////////////////////////////////////////////////////////
                0112  3709  ;+
                0112  3710  ; Test the IRD1 ROMS with the FPA present and enabled.
```

```
                    0112  3711 ;-
                    0112  3712          LOADREG         WDREG,8$,,L             ; LOAD DATA TO ENABLE THE FPA
                    011A  3713          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                    011C  3714          FETCH           <^X10>                  ; ENABLE THE FPA
                    0121  3715          BRSTCLK         <^X14>,INH              ; AND READ BACK FPA ENABLE
                    0125  3716          CMPREGMSK       CSTRP,4$,,5$,,W         ; IS THE FPA ENABLED?
                    0131  3717          IFERROR         ,,<FPA>                 ; IF NOT, SIGNAL AN ERROR
                    0137  3718          EXPUTRAP                                ; ELSE EXPECT NON-DCS ADDRESSES
                    0139  3719          LOOP            I,1,256                 ; LOOP FOR 256 OPCODES
                    0140  3720          SAVEINDEX       I,9$                    ; SAVE I-1 TO FORM AN OPCODE
                    0145  3721          LOADREG         WDREG,9$,,L             ; LOAD WDREG WITH THE OPCODE
                    014D  3722          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                    014F  3723          FETCH           0                       ; EXECUTE THE IRD1 ADDRESS
                    0154  3724          BRSTCLK         <^X0F>,INH              ; TEST MICROCODE
                    0158  3725          CMPREGMSK       CSTRP,11$,I,6$,,W       ; IS THE CS ADDRESS CORRECT?
                    0164  3726          IFERROR         1,<IRD>,<MIC>           ; IF NOT, SIGNAL AN ERROR
                    016B  3727          ENDLOOP         I                       ; CONTINUE THE LOOP
                    016E  3728          SKIP                                    ; SKIP TO THE NEXT TEST
                    0175  3729
                    0175  3730 BEGIN_TEST_DATA
                    0175
                    0175                ;-------------------------------------------------------------------------
                    0175  3731
                1802 0175  3732 1$:     .WORD   ^X1802                  ; ADDRESS GENERATED WHEN FPA IS PRESENT
                3FFE 0177  3733 2$:     .WORD   ^X3FFE                  ; ADDRESS MASK FOR FPA PRESENCE TEST
                    0179  3734
                1801 0179  3735 3$:     .WORD   ^X1801                  ; ADDRESS GENERATED WHEN THE FPA IS DISABLED
                1800 017B  3736 4$:     .WORD   ^X1800                  ; ADDRESS GENERATED WHEN THE FPA IS ENABLED
                3FFD 017D  3737 5$:     .WORD   ^X3FFD                  ; ADDRESS MASK FOR FPA ENABLE TEST
                    017F  3738
                3FFF 017F  3739 6$:     .WORD   ^X3FFF                  ; ADDRESS COMPARISON MASK
                    0181  3740
            00000000 0181  3741 7$:     .LONG   0                       ; DATA TO DISABLE THE FPA
            00000020 0185  3742 8$:     .LONG   ^X00000020              ; DATA TO ENABLE THE FPA
                    0189  3743
            00000000 0189  3744 9$:     .LONG   0                       ; LOCATION TO HOLD LOOP COUNTER
                    018D  3745
                    018D  3746 ;
                    018D  3747 ; Expected CS Addresses for Subtests 1 and 2
                    018D  3748 ;
                    018D  3749 ; The following is a table of all the expected control store addresses for an
                    018D  3750 ; IRD1 with the FPA disabled. The hex numbers in the comments to the right
                    018D  3751 ; indicate the range of loop counter values for each row of address words.
                    018D  3752 ;
                    018D  3753
03B0 03B0 03B0 03B0 03B0 03B0 03B0  018D  3754 10$:    .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 01 - 07
03B0 03B0 0278 03B0 0378 0378 03B0  019B  3755         .WORD   ^X03B0,^X0378,^X0378,^X03B0,^X0278,^X03B0,^X03B0 ; 08 - 0E
03B0 03B0 03B0 03B0 03B0 03B0 03B0  01A9  3756         .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 0F - 15
03B0 03B0 03B0 03B0 03B0 03B0 03B0  01B7  3757         .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 16 - 1C
0380 0380 0380 03B0 03B0 03B0 03B0  01C5  3758         .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X0380,^X0380,^X0380 ; 1D - 23
0330 0328 0350 0378 0380 0378 0380  01D3  3759         .WORD   ^X0380,^X0378,^X0380,^X0378,^X0350,^X0328,^X0330 ; 24 - 2A
03B0 0328 0328 0330 0328 0338 0338  01E1  3760         .WORD   ^X0338,^X0338,^X0328,^X0330,^X0328,^X0328,^X03B0 ; 2B - 31
0380 0358 0380 0380 03B0 03B0 03B0  01EF  3761         .WORD   ^X03B0,^X03B0,^X03B0,^X0380,^X0380,^X0358,^X0380 ; 32 - 38
03B0 03B0 03B0 0340 0340 0348 0388  01FD  3762         .WORD   ^X0388,^X0348,^X0340,^X0340,^X03B0,^X03B0,^X03B0 ; 39 - 3F
03B0 03B0 03B0 03B0 03B0 03B0 03B0  020B  3763         .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 40 - 46
```

```
03B0 03B0 03B0 03B0 03B0 03B0 03B0  0219  3764          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 47 - 4D
03B0 03B0 03B0 03B0 03B0 03B0 03B0  0227  3765          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 4E - 54
03B0 03B0 03B0 03B0 03B0 0200 03B0  0235  3766          .WORD   ^X03B0,^X0200,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 55 - 5B
03B0 03B0 03B0 03B0 03B0 03B0 03B0  0243  3767          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 5C - 62
03B0 03B0 03B0 03B0 03B0 03B0 03B0  0251  3768          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 63 - 69
03B0 03B0 03B0 03B0 03B0 03B0 03B0  025F  3769          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 6A - 70
03B0 0220 03B0 03B0 03B0 03B0 03B0  026D  3770          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X0220,^X03B0 ; 71 - 77
03B0 03B0 03B0 03B0 03B0 03B0 03B0  027B  3771          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 78 - 7E
03B0 03B0 03B0 03B0 03B0 03B0 03B0  0289  3772          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 7F - 85
03B0 03B0 03B0 03B0 03B0 03B0 03B0  0297  3773          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 86 - 8C
03B0 03B0 03B0 03B0 03B0 03B0 03B0  02A5  3774          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 8D - 93
03B0 03B0 03B0 03B0 03B0 03B0 03B0  02B3  3775          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 94 - 9A
03B0 03B0 03B0 03B0 03B0 03B0 03B0  02C1  3776          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 9B - A1
03B0 03B0 03B0 03B0 03B0 03B0 03B0  02CF  3777          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; A2 - A8
03B0 03B0 03B0 03B0 03B0 03B0 03B0  02DD  3778          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; A9 - AF
03B0 03B0 03B0 03B0 03B0 03B0 03B0  02EB  3779          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; B0 - B6
03B0 03B0 03B0 03B0 03B0 03B0 03B0  02F9  3780          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; B7 - BD
03B0 03B0 03B0 03B0 03B0 03B0 03B0  0307  3781          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; BE - C4
03B0 03B0 03B0 03B0 03B0 03B0 03B0  0315  3782          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; C5 - CB
03B0 03B0 03B0 03B0 03B0 03B0 03B0  0323  3783          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; CC - D2
03B0 03B0 03B0 03B0 03B0 03B0 03B0  0331  3784          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; D3 - D9
03B0 03B0 03B0 03B0 03B0 03B0 03B0  033F  3785          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; DA - E0
03B0 03B0 03B0 03B0 03B0 03B0 03B0  034D  3786          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; E1 - E7
03B0 03B0 03B0 03B0 03B0 03B0 03B0  035B  3787          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; E8 - EE
03B0 03B0 03B0 03B0 03B0 03B0 03B0  0369  3788          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; EF - F5
03B0 03B0 03B0 0380 03B0 03B0 03B0  0377  3789          .WORD   ^X03B0,^X03B0,^X03B0,^X0380,^X03B0,^X03B0,^X03B0 ; F6 - FC
          03B0 03B0 03F0 02C8       0385  3790          .WORD   ^X02C8,^X03F0,^X03B0,^X03B0                       ; FD - 100
                                    038D  3791
                                    038D  3792 ;
                                    038D  3793 ; Expected CS Addresses for Subtest 3
                                    038D  3794 ;
                                    038D  3795 ; The following is a table of all the expected control store addresses for an
                                    038D  3796 ; IRD1 with the FPA enabled. The hex numbers in the comments to the right
                                    038D  3797 ; indicate the range of loop counter values for each row of address words.
                                    038D  3798 ;
                                    038D  3799
03B0 03B0 03B0 03B0 03B0 03B0 03B0  038D  3800 11$:      .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 01 - 07
03B0 03B0 0278 03B0 0378 0378 03B0  039B  3801          .WORD   ^X03B0,^X0378,^X0378,^X03B0,^X0278,^X03B0,^X03B0 ; 08 - 0E
03B0 03B0 03B0 03B0 03B0 03B0 03B0  03A9  3802          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 0F - 15
03B0 03B0 03B0 03B0 03B0 03B0 03B0  03B7  3803          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 16 - 1C
0380 0380 0380 03B0 03B0 03B0 03B0  03C5  3804          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X0380,^X0380,^X0380 ; 1D - 23
0330 0328 0350 0378 0380 0378 0380  03D3  3805          .WORD   ^X0380,^X0378,^X0380,^X0378,^X0350,^X0328,^X0330 ; 24 - 2A
03B0 0328 0328 0330 0328 0338 0338  03E1  3806          .WORD   ^X0338,^X0338,^X0328,^X0330,^X0328,^X0328,^X03B0 ; 2B - 31
0380 0358 0380 0380 03B0 03B0 03B0  03EF  3807          .WORD   ^X03B0,^X03B0,^X03B0,^X0380,^X0380,^X0358,^X0380 ; 32 - 38
03B0 03B0 03B0 0340 0340 0348 0388  03FD  3808          .WORD   ^X0388,^X0348,^X0340,^X0340,^X03B0,^X03B0,^X03B0 ; 39 - 3F
03B0 03B0 03B0 03B0 03B0 03B0 03B0  040B  3809          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 40 - 46
03B0 03B0 03B0 03B0 03B0 03B0 03B0  0419  3810          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 47 - 4D
03B0 03B0 03B0 03B0 03B0 03B0 03B0  0427  3811          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 4E - 54
03B0 03B0 03B0 03B0 03B0 0240 03B0  0435  3812          .WORD   ^X03B0,^X0240,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 55 - 5B
03B0 03B0 03B0 03B0 03B0 03B0 03B0  0443  3813          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 5C - 62
03B0 03B0 03B0 03B0 03B0 03B0 03B0  0451  3814          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 63 - 69
03B0 03B0 03B0 03B0 03B0 03B0 03B0  045F  3815          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 6A - 70
03B0 0258 03B0 03B0 03B0 03B0 03B0  046D  3816          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X0258,^X03B0 ; 71 - 77
03B0 03B0 03B0 03B0 03B0 03B0 03B0  047B  3817          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 78 - 7E
03B0 03B0 03B0 03B0 03B0 03B0 03B0  0489  3818          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 7F - 85
```

```
 03B0 03B0 03B0 03B0 03B0 03B0 03B0  0497  3819          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 86 - 8C
 03B0 03B0 03B0 03B0 03B0 03B0 03B0  04A5  3820          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 8D - 93
 03B0 03B0 03B0 03B0 03B0 03B0 03B0  04B3  3821          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 94 - 9A
 03B0 03B0 03B0 03B0 03B0 03B0 03B0  04C1  3822          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; 9B - A1
 03B0 03B0 03B0 03B0 03B0 03B0 03B0  04CF  3823          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; A2 - A8
 03B0 03B0 03B0 03B0 03B0 03B0 03B0  04DD  3824          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; A9 - AF
 03B0 03B0 03B0 03B0 03B0 03B0 03B0  04EB  3825          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; B0 - B6
 03B0 03B0 03B0 03B0 03B0 03B0 03B0  04F9  3826          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; B7 - BD
 03B0 03B0 03B0 03B0 03B0 03B0 03B0  0507  3827          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; BE - C4
 03B0 03B0 03B0 03B0 03B0 03B0 03B0  0515  3828          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; C5 - CB
 03B0 03B0 03B0 03B0 03B0 03B0 03B0  0523  3829          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; CC - D2
 03B0 03B0 03B0 03B0 03B0 03B0 03B0  0531  3830          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; D3 - D9
 03B0 03B0 03B0 03B0 03B0 03B0 03B0  053F  3831          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; DA - E0
 03B0 03B0 03B0 03B0 03B0 03B0 03B0  054D  3832          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; E1 - E7
 03B0 03B0 03B0 03B0 03B0 03B0 03B0  055B  3833          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; E8 - EE
 03B0 03B0 03B0 03B0 03B0 03B0 03B0  0569  3834          .WORD   ^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0,^X03B0 ; EF - F5
 03B0 03B0 03B0 0380 03B0 03B0 03B0  0577  3835          .WORD   ^X03B0,^X03B0,^X03B0,^X0380,^X03B0,^X03B0,^X03B0 ; F6 - FC
                03B0 03B0 03F0 02C8  0585  3836          .WORD   ^X02C8,^X03F0,^X03B0,^X03B0                      ; FD - 100
                                     058D  3837
                                     058D  3838 BEGIN_CPU_DATA
                                     0002  3839
                                     0002  3840 DCS_DATA
                                     0001  3841
U 00, 7700,C800,0364,0300,0470,1801  0001  3842 ;* 00   NOP                                  ; GET UP TO SPEED
U 01, 5700,C840,0364,0304,0470,1802  000C  3846 ;* 01   R[TEMP1]_RDM                         ; LOAD THE OPCODE INTO RTEMP1
U 02, 7700,F800,57D7,D7FF,8470,1803  0017  3850 ;* 02   LONLIT_[50505000]                   ; LOAD THE OPERANDS INTO THE LONLIT REG
U 03, 7700,8800,5BE5,03D4,0470,1804  0022  3854 ;* 03   Q_R[LONLIT]                         ; MOVE THE OPERANDS TO THE Q REGISTER
U 04, 7700,0841,00A4,0304,0470,1805  002D  3858 ;* 04   R[TEMP1]_M[TEMP1].OR.Q              ; OR THE OPCODE WITH THE OPERANDS
U 05, 7700,4800,5BE4,03D8,4A70,1806  0038  3862 ;* 05   VA_R[ZERO]                         ; ZERO THE VA
U 06, 7701,C800,5BE4,0204,5590,1807  0043  3866 ;* 06   WRITE.LONG R[TEMP1]                ; WRITE THE MODEL INSTRUCTION TO CACHE
U 07, 7700,8800,5BE4,0304,5470,1808  004E  3870 ;* 07   WDR_UNROT(R[TEMP1])               ; CMI OFF SO HOLD DATA FOR SECOND CYCLE
U 08, 7700,C800,0364,0300,4470,1809  0059  3874 ;* 08   VA_VA+4                           ; INCREMENT THE VA
U 09, 7701,0800,5BE4,02D8,5590,180A  0064  3878 ;* 09   WRITE.LONG R[ZERO]                ; VALIDATE NEXT LOCATION FOR XB1
U 0A, 7700,C800,5BE4,03D8,5470,180B  006F  3882 ;* 0A   WDR_UNROT(R[ZERO])               ; CMI OFF SO HOLD DATA FOR SECOND CYCLE
U 0B, 7700,C800,5BE4,03D8,4870,180C  007A  3886 ;* 0B   PC_R[ZERO]                       ; START INSTRUCTION PREFETCH, LOAD XB'S
U 0C, 7701,0857,5924,0202,0470,180D  0085  3890 ;* 0C   R[TEMP0]_M[XB.PC_PC+I],ISIZE[LONG] ; LOAD RTEMP0 WITH XB0
U 0D, 7700,4800,5BE4,03D8,4870,180E  0090  3894 ;* 0D   PC_R[ZERO]                       ; RESET THE PC AND RELOAD THE XB'S
U 0E, 7700,0800,0364,1300,0470,180F  009B  3898 ;* 0E   BUT/IRD1                         ; EXECUTE AN INSTRUCTION DECODE
U 0F, 7300,C800,0364,0300,0470,1810  00A6  3902 ;* 0F   STOP.CLOCK                       ; HALT
                                     00B1  3906
U 10, 7700,4800,0364,0300,0470,1811  00B1  3907 ;* 10   NOP                              ; GET CPU UP TO SPEED
U 11, 5700,4800,0364,0300,2A70,1812  00BC  3911 ;* 11   WCTRL/FPA.ENABLE_WB5,WB_RDM      ; WRITE THE FPA'S ENABLE BIT
U 12, 7700,C800,0364,0300,0470,1813  00C7  3915 ;* 12   NOP                              ; NOP TO GET CPU UP TO SPEED
U 13, 7700,4800,0364,C300,0470,1800  00D2  3919 ;* 13   BUT/NO.FPA,NEXT/1800            ; CHECK FPA STATUS
U 14, 7300,C800,0364,0300,0470,1815  00DD  3923 ;* 14   STOP.CLOCK                       ; HALT
                                     00E8  3927
U 15, 7700,C800,0364,0300,0470,1816  00E8  3928 ;* 15   NOP                              ; NOP TO GET CPU UP TO SPEED
U 16, 7700,B800,7BFF,FFFF,8470,1817  00F3  3932 ;* 16   LONLIT_[08000000]               ; LOAD LONGLIT WITH DATA TO SET FPD
U 17, 7700,8800,5BF4,03D4,0070,1818  00FE  3936 ;* 17   PSL_R[LONLIT]                    ; SET FPD BIT IN PSL
U 18, 7700,7800,7F7F,7F7F,0470,1819  0109  3940 ;* 18   LONLIT_[01010101]               ; LOAD LONGLIT WITH 4 NOP INSTRUCTIONS
U 19, 7700,C800,5BE4,03D8,4A70,181A  0114  3944 ;* 19   VA_R[ZERO]                       ; ZERO THE VA
U 1A, 7701,0800,5BE4,02D4,5590,181B  011F  3948 ;* 1A   WRITE.LONG R[LONLIT]            ; WRITE NOPS TO CACHE FOR XB0
U 1B, 7700,C800,5BE4,03D4,5470,181C  012A  3952 ;* 1B   WDR_UNROT(R[LONLIT])           ; CMI OFF SO HOLD FOR SECOND CYCLE
U 1C, 7700,C800,0364,0300,4470,181D  0135  3956 ;* 1C   VA_VA+4                         ; INCREMENT THE VA
U 1D, 7701,0800,5BE4,02D4,5590,181E  0140  3960 ;* 1D   WRITE.LONG R[LONLIT]            ; VALIDATE NEXT LOCATION FOR XB1
```

```
U 1E, 7700,C800,5BE4,03D4,5470,181F  014B  3964 ;% 1E   WDR_UNROT(R[LONLIT])          ; CMI OFF SO HOLD FOR SECOND CYCLE
U 1F, 7700,4800,5BE4,03D8,4870,1820  0156  3968 ;% 1F   PC_R[ZERO]                    ; ZERO THE PC TO LOAD XB'S
U 20, 7700,4800,0364,1700,0470,1821  0161  3972 ;% 20   IRD1TEST                     ; IRD1 TO SET FPA FOR NON-FPA OPCODE
U 21, 7300,4800,0364,0300,0470,1822  016C  3976 ;% 21   STOP.CLOCK                   ; HALT
                                     0177  3980
                                     0177  3981 RTEMP_DATA
                                     0001  3982
                           00000000  0001  3983         .LONG   0       ; RTEMP0 HOLDS THE CONTENTS OF XB0
                           00000000  0005  3984         .LONG   0       ; RTEMP1 HOLDS THE INSTRUCTION OPCODE
                                     0009  3985
                                     0009  3986 END_CPU_DATA
                                     058D  3987
                                     058D  3988 END_TEST_DATA
                                     058D
                                     058D       ;-------------------------------------------------------------------------
                                     058D  3989
                                     058D  3990 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
                        002B        .SBTTL  "TEST   0C5     IRD1 Instruction Address Decode Test 4
                        002B  3992 ;*******************************************************************************
                        002B  3993 ;++
                        002B  3994 ;
                        002B  3995 ; FUNCTIONAL DESCRIPTION
                        002B  3996 ;
                        002B  3997 ;       This test will examine the signal ROM OS INH H in the ROMS that
                        002B  3998 ;       implement a native mode IRD1 when the First Part Done (FPD) bit of the
                        002B  3999 ;       PSL is set to a one.
                        002B  4000 ;
                        002B  4001 ; TEST ALGORITHM
                        002B  4002 ;
                        002B  4003 ;       This test is implemented in three subtests. The first subtest examines
                        002B  4004 ;       the IRD1 ROMS when the CPU's Floating Point Accelerator (FPA) is not
                        002B  4005 ;       present. The second subtest examines the ROMS when there is an FPA
                        002B  4006 ;       present and it has been disabled. The third subtest examines the ROMS
                        002B  4007 ;       when the FPA is present and enabled. The test algorithm executes as
                        002B  4008 ;       follows:
                        002B  4009 ;
                        002B  4010 ;        1. Initialize the CPU. This will clear the FPD bit in the PSL.
                        002B  4011 ;        2. Execute the following microcode to set the FPD bit in the PSL and
                        002B  4012 ;           to prepare an FPA to be enabled or disabled:
                        002B  4013 ;                1. NOP to get the CPU up to speed.
                        002B  4014 ;                2. Load the data needed to set the FPD bit into longlit.
                        002B  4015 ;                3. Set the FPD bit in the PSL.
                        002B  4016 ;                4. Load four NOP opcodes into the longlit register for XB0.
                        002B  4017 ;                5. Zero the VA.
                        002B  4018 ;                6. Write the NOPS for XB0 into cache at address 0.
                        002B  4019 ;                7. Increment the VA by 4.
                        002B  4020 ;                8. Validate the next location for XB1 prefetch by writing
                        002B  4021 ;                   NOPS to cache at address 4.
                        002B  4022 ;                9. Zero the PC to initiate the instruction prefetch that
                        002B  4023 ;                   writes XB0 and XB1 from cache.
                        002B  4024 ;               10. Execute an IRD1TEST cycle to decode the NOP in XB0. This
                        002B  4025 ;                   will intialize the FPA to allow it to be enabled/disabled.
                        002B  4026 ;               11. STOP.CLOCK to halt.
                        002B  4027 ;        3. Execute the following microcode to test for an FPA presence:
                        002B  4028 ;               12. NOP to get the CPU up to speed.
                        002B  4029 ;               13. BUT/NO.FPA,NEXT/1800
                        002B  4030 ;               14. STOP.CLOCK to halt.
                        002B  4031 ;        4. Perform a masked comparison of the address in the Control Store
                        002B  4032 ;           Address Trace Register (CSTRP) against the expected address for the
                        002B  4033 ;           FPA not present.
                        002B  4034 ;        5. If the two do not match, then there is an FPA, so branch to
                        002B  4035 ;           Subtest 2, otherwise continue on to Subtest 1.
                        002B  4036 ;
                        002B  4037 ;       Subtest 1
                        002B  4038 ;
                        002B  4039 ;        6. Loop on error from here.
                        002B  4040 ;        7. Re-execute the microcode in step 3 to test the FPA enable bit.
                        002B  4041 ;        8. Perform a masked comparison of the CSTRP with the expected address
                        002B  4042 ;           for the FPA disabled.
                        002B  4043 ;        9. If the two do not match then there is a problem with the FPA/CPU
                        002B  4044 ;           interface so signal an error.
                        002B  4045 ;       10. Prepare the RDM to expect non-DCS addresses.
```

```
002B  4046 ;       11. Prepare a loop on I for 256 VAX opcodes.
002B  4047 ;       12. Save the loop counter minus one in the RDM memory.
002B  4048 ;       13. Load the WDREG with the number saved in RDM memory.
002B  4049 ;       14. Loop on error from here.
002B  4050 ;       15. Execute the following test microcode:
002B  4051 ;                15. NOP to get the CPU up to speed.
002B  4052 ;                16. Load the opcode to be tested from the WDREG to RTEMP1.
002B  4053 ;                17. Load the operands into the longlit register.
002B  4054 ;                18. Move the operands from the longlit register into the
002B  4055 ;                    Q register.
002B  4056 ;                19. Form the test instruction by ORing the opcode in RTEMP1
002B  4057 ;                    (MTEMP1=RTEMP1) with the operands in the Q register and
002B  4058 ;                    write the result back to RTEMP1.
002B  4059 ;                20. Zero the VA.
002B  4060 ;                21. Write the test instruction in RTEMP1 to cache at address 0.
002B  4061 ;                22. Increment the VA by 4.
002B  4062 ;                23. Validate the next location for XB1 prefetch by writing
002B  4063 ;                    zeros to cache at address 4.
002B  4064 ;                24. Zero the PC to initiate the instruction prefetch that
002B  4065 ;                    writes XB0 and XB1 from cache.
002B  4066 ;                25. Write the contents of XB0 to RTEMP0. This will cause the
002B  4067 ;                    PC to increment by 4 and can change the contents of XB0.
002B  4068 ;                26. Rezero the PC to reload XB0 and XB1.
002B  4069 ;                27. Execute a BUT/IRD1 to decode the instruction in XB0 and
002B  4070 ;                    strobe the VBUS.
002B  4071 ;                28. Halt.
002B  4072 ;       16. Compare the VBUS bit 1E (LOAD OSR L) with the expected value.
002B  4073 ;       17. If the two are not equal, signal an error.
002B  4074 ;       18. Continue to loop on I for the remaining VAX opcodes.
002B  4075 ;       19. Skip to the next test.
002B  4076 ;
002B  4077 ;       Subtests 2 and 3
002B  4078 ;
002B  4079 ;       21. Load the WDREG with a longword to perform an FPA disable for
002B  4080 ;           Subtest 2 or an FPA enable for Subtest 3.
002B  4081 ;       21. Loop on error from here.
002B  4082 ;       22. Execute the following microcode to disable the FPA for Subtest 2 or
002B  4083 ;           to enable the FPA for Subtest 3 and to also check the result:
002B  4084 ;                29. NOP to get the CPU up to speed.
002B  4085 ;                30. WCTRL/FPAENABLE_WB5,WB_RDM to write the FPA's enable bit
002B  4086 ;                    with bit 5 of the RDM's WDREG.
002B  4087 ;                31. NOP
002B  4088 ;                32. BUT/NO.FPA,NEXT/1800 to test for FPA enabled.
002B  4089 ;                33. STOP.CLOCK to halt.
002B  4090 ;       23. Perform a masked comparison of the CSTRP with the expected address
002B  4091 ;           for the FPA disabled (Subtest 2) or the FPA enabled (Subtest 3).
002B  4092 ;       24. If the two do not match then there is a problem with the FPA/CPU
002B  4093 ;           interface so signal an error.
002B  4094 ;       25. Prepare the RDM to expect non-DCS addresses.
002B  4095 ;       26. Prepare a loop on I for 256 VAX opcodes.
002B  4096 ;       27. Save the loop counter minus one in the RDM memory.
002B  4097 ;       28. Load the WDREG with the number saved in RDM memory.
002B  4098 ;       29. Loop on error from here.
002B  4099 ;       30. Execute the same test microcode as in Subtest 1 step 15.
002B  4100 ;       31. Compare the VBUS bit 1E (LOAD OSR L) with the expected value.
```

```
002B  4101 ;       32. If the two are not equal, signal an error.
002B  4102 ;       33. Continue to loop on I for the remaining VAX opcodes.
002B  4103 ;       34. If this is Subtest 2 then continue on to Subtest 3.
002B  4104 ;           If this is Subtest 3 then skip to the next test.
002B  4105 ;
002B  4106 ; TEST PATTERNS
002B  4107 ;
002B  4108 ;       For this test the VAX machine opcodes are generated by a simple count
002B  4109 ;       pattern. The opcodes are based on the 256 base machine instructions
002B  4110 ;       as specified in the VAX architecture handbook. The generated opcodes
002B  4111 ;       are written to the CPU and combined with three bytes of operand
002B  4112 ;       data to form a VAX machine instruction. The operands all specify R0 in
002B  4113 ;       register mode and are loaded from the CPU's long literal register.
002B  4114 ;
002B  4115 ;       Also required are two tables of expected VBUS values in bit 1E for
002B  4116 ;       each IRD1 Instruction Decode Cycle. One table contains the expected
002B  4117 ;       VBUS data for an IRD1 when the FPA has been disabled, the other for
002B  4118 ;       when it is enabled.  The expected VBUS data was obtained from the
002B  4119 ;       Native Mode Ird Rom Definition  in the CPU microcode.
002B  4120 ;
002B  4121 ; LOGIC DESCRIPTION
002B  4122 ;
002B  4123 ;       The CCS ROMS are fed addresses from ROMS on the DPM during instruction
002B  4124 ;       decode. The DPM ROMS are addressed by bits <07:00> of the execution
002B  4125 ;       buffer. These bits form the opcode for the instruction. For zero and
002B  4126 ;       certain single operand instructions the addresses generated by the ROMS
002B  4127 ;       will force a jump to execution flows in the microcode. For the rest,
002B  4128 ;       operand specifier evaulation addresses will be generated.
002B  4129 ;
002B  4130 ; ASSUMPTIONS
002B  4131 ;
002B  4132 ;       This test assumes the following logic has already been tested; the
002B  4133 ;       CPU's data path, the cache, the XB, and the IR and OSR.
002B  4134 ;
002B  4135 ; ERROR DESCRIPTION
002B  4136 ;
002B  4137 ;       On error this test can type out the following information:
002B  4138 ;
002B  4139 ;       Failing gate arrays:    IRD
002B  4140 ;       Failing modules:        FPA, MIC
002B  4141 ;
002B  4142 ;       The following data will also by typed:
002B  4143 ;
002B  4144 ;       Expected VBUS value 00000xnn            where   x  = VBUS bit value
002B  4145 ;       Received VBUS value 00000xnn                    nn = VBUS bit position
002B  4146 ;       Loop count (index I)
002B  4147 ;       The instruction being decoded from XB0 (RTEMP0)
002B  4148 ;
002B  4149 ;--
002B  4150 ;*****************************************************************************
002B  4151
002B  4152         INITIALIZE                                  ; INITIALIZE THE CPU
002D  4153         FETCH           <^X15>                      ; SET FPD BIT IN PSL AND PREP.
0032  4154         BRSTCLK         <^X21>                      ; AN FPA TO BE ENABLED/DISABLED
0036  4155         FETCH           <^X12>                      ; EXECUTE A BUT/NO.FPA TO TEST
```

```
003B  4156          BRSTCLK         <^X14>                          ; FOR AN FPA PRESENT
003F  4157          CMPREGMSK       CSTRP,1$,,2$,,W                 ; CHECK THE BUT/NO.FPA ADDRESS
004B  4158          SKIP            100$,ONERROR                    ; SKIP IF THE FPA IS PRESENT
0052  4159
0052  4160          SUBTEST         1
0055
0055          ;////////////////////////////////////////////////////////////////////////////////
0055  4161  ;+
0055  4162  ; Test the IRD1 ROMS when the FPA is not present.
0055  4163  ;-
0055  4164          ERRLOOP                                         ; LOOP ON ERROR FROM HERE
0057  4165          FETCH           <^X12>                          ; EXECUTE A BUT/NO.FPA TO TEST
005C  4166          BRSTCLK         <^X14>,INH                      ; FOR THE FPA DISABLED
0060  4167          CMPREGMSK       CSTRP,3$,,5$,,W                 ; IS THE FPA DISABLED?
006C  4168          IFERROR         ,,<FPA>                         ; IF NOT, SIGNAL AN ERROR
0072  4169          EXPUTRAP                                        ; EXPECT NON-DCS ADDRESSES
0074  4170          LOOP            I,1,256                         ; LOOP FOR 256 OPCODES
007B  4171          SAVEINDEX       I,9$                            ; SAVE I-1 TO FORM AN OPCODE
0080  4172          LOADREG         WDREG,9$,,L                     ; LOAD WDREG WITH THE OPCODE
0088  4173          ERRLOOP                                         ; LOOP ON ERROR FROM HERE
008A  4174          FETCH           0                               ; EXECUTE THE IRD1 ADDRESS
008F  4175          BRSTCLK         <^X0F>,INH                      ; TEST MICROCODE
0093  4176          CMPVBUS         10$,I                           ; IS VBUS BIT 1E CORRECT?
0098  4177          IFERROR         1,<IRD>,<MIC>                   ; IF NOT, SIGNAL AN ERROR
009F  4178          ENDLOOP         I                               ; CONTINUE THE LOOP
00A2  4179          SKIP                                            ; SKIP TO THE NEXT TEST
00A9  4180
00A9  4181  100$:   SUBTEST         2
00AC
00AC          ;////////////////////////////////////////////////////////////////////////////////
00AC  4182  ;+
00AC  4183  ; Test the IRD1 ROMS with the FPA present but not enabled.
00AC  4184  ;-
00AC  4185          LOADREG         WDREG,7$,,L                     ; LOAD DATA TO DISABLE THE FPA
00B4  4186          ERRLOOP                                         ; LOOP ON ERROR FROM HERE
00B6  4187          FETCH           <^X10>                          ; EXECUTE AN FPA DISABLE
00BB  4188          BRSTCLK         <^X14>,INH                      ; AND READ BACK THE FPA STATUS
00BF  4189          CMPREGMSK       CSTRP,3$,,5$,,W                 ; IS THE FPA DISABLED?
00CB  4190          IFERROR         ,,<FPA>                         ; IF NOT, SIGNAL AN ERROR
00D1  4191          EXPUTRAP                                        ; EXPECT NON-DCS ADDRESSES
00D3  4192          LOOP            I,1,256                         ; LOOP FOR 256 OPCODES
00DA  4193          SAVEINDEX       I,9$                            ; SAVE I-1 TO FORM AN OPCODE
00DF  4194          LOADREG         WDREG,9$,,L                     ; LOAD WDREG WITH THE OPCODE
00E7  4195          ERRLOOP                                         ; LOOP ON ERROR FROM HERE
00E9  4196          FETCH           0                               ; EXECUTE THE IRD1 ADDRESS
00EE  4197          BRSTCLK         <^X0F>,INH                      ; TEST MICROCODE
00F2  4198          CMPVBUS         10$,I                           ; IS VBUS BIT 1E CORRECT?
00F7  4199          IFERROR         1,<IRD>,<MIC>                   ; IF NOT, SIGNAL AN ERROR
00FE  4200          ENDLOOP         I                               ; CONTINUE THE LOOP
0101  4201
0101  4202          SUBTEST         3
0104
0104          ;////////////////////////////////////////////////////////////////////////////////
0104  4203  ;+
0104  4204  ; Test the IRD1 ROMS with the FPA present and enabled.
```

```
                        0104  4205 ;-
                        0104  4206          LOADREG         WDREG,8$,,L             ; LOAD DATA TO ENABLE THE FPA
                        010C  4207          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        010E  4208          FETCH           <^X10>                  ; ENABLE THE FPA
                        0113  4209          BRSTCLK         <^X14>,INH              ; AND READ BACK FPA ENABLE
                        0117  4210          CMPREGMSK       CSTRP,4$,,5$,,W         ; IS THE FPA ENABLED?
                        0123  4211          IFERROR         ,,<FPA>                 ; IF NOT, SIGNAL AN ERROR
                        0129  4212          EXPUTRAP                                ; ELSE EXPECT NON-DCS ADDRESSES
                        012B  4213          LOOP            I,1,256                 ; LOOP FOR 256 OPCODES
                        0132  4214          SAVEINDEX       I,9$                    ; SAVE I-1 TO FORM AN OPCODE
                        0137  4215          LOADREG         WDREG,9$,,L             ; LOAD WDREG WITH THE OPCODE
                        013F  4216          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        0141  4217          FETCH           0                       ; EXECUTE THE IRD1 ADDRESS
                        0146  4218          BRSTCLK         <^X0F>,INH              ; TEST MICROCODE
                        014A  4219          CMPVBUS         11$,I                   ; IS VBUS BIT 1E CORRECT?
                        014F  4220          IFERROR         1,<IRD>,<MIC>           ; IF NOT, SIGNAL AN ERROR
                        0156  4221          ENDLOOP         I                       ; CONTINUE THE LOOP
                        0159  4222          SKIP                                    ; SKIP TO THE NEXT TEST
                        0160  4223
                        0160  4224 BEGIN_TEST_DATA
                        0160
                        0160       ;-----------------------------------------------------------------------
                        0160  4225
                1802    0160  4226 1$:      .WORD   ^X1802          ; ADDRESS GENERATED WHEN FPA IS PRESENT
                3FFE    0162  4227 2$:      .WORD   ^X3FFE          ; ADDRESS MASK FOR FPA PRESENCE TEST
                        0164  4228
                1801    0164  4229 3$:      .WORD   ^X1801          ; ADDRESS GENERATED WHEN THE FPA IS DISABLED
                1800    0166  4230 4$:      .WORD   ^X1800          ; ADDRESS GENERATED WHEN THE FPA IS ENABLED
                3FFD    0168  4231 5$:      .WORD   ^X3FFD          ; ADDRESS MASK FOR FPA ENABLE TEST
                        016A  4232
                3FFF    016A  4233 6$:      .WORD   ^X3FFF          ; ADDRESS COMPARISON MASK
                        016C  4234
            00000000    016C  4235 7$:      .LONG   0               ; DATA TO DISABLE THE FPA
            00000020    0170  4236 8$:      .LONG   ^X00000020      ; DATA TO ENABLE THE FPA
                        0174  4237
            00000000    0174  4238 9$:      .LONG   0               ; LOCATION TO HOLD LOOP COUNTER
                        0178  4239
                        0178  4240 ;
                        0178  4241 ; Expected VBUS bit values for Subtests 1 and 2
                        0178  4242 ;
                        0178  4243 ; The following is a table of all the VBUS bit values for an IRD1 with the FPA
                        0178  4244 ; disabled. The hex numbers in the comments to the right indicate the range of
                        0178  4245 ; loop counter values for each row of address words.
                        0178  4246 ;
                        0178  4247
                  01    0178  4248 10$:     .BYTE   ^X01
                        0179  4249          VBUSDATA        <^X1E>,1                ;00
                  01    017A  4250          .BYTE   ^X01
                        017B  4251          VBUSDATA        <^X1E>,1                ;01
                  01    017C  4252          .BYTE   ^X01
                        017D  4253          VBUSDATA        <^X1E>,1                ;02
                  01    017E  4254          .BYTE   ^X01
                        017F  4255          VBUSDATA        <^X1E>,1                ;03
                  01    0180  4256          .BYTE   ^X01
                        0181  4257          VBUSDATA        <^X1E>,1                ;04
```

```
                    01  0182  4258          .BYTE   ^X01
                        0183  4259          VBUSDATA        <^X1E>,1                ;05
                    01  0184  4260          .BYTE   ^X01
                        0185  4261          VBUSDATA        <^X1E>,1                ;06
                    01  0186  4262          .BYTE   ^X01
                        0187  4263          VBUSDATA        <^X1E>,1                ;07
                    01  0188  4264          .BYTE   ^X01
                        0189  4265          VBUSDATA        <^X1E>,1                ;08
                    01  018A  4266          .BYTE   ^X01
                        018B  4267          VBUSDATA        <^X1E>,1                ;09
                    01  018C  4268          .BYTE   ^X01
                        018D  4269          VBUSDATA        <^X1E>,1                ;0A
                    01  018E  4270          .BYTE   ^X01
                        018F  4271          VBUSDATA        <^X1E>,1                ;0B
                    01  0190  4272          .BYTE   ^X01
                        0191  4273          VBUSDATA        <^X1E>,1                ;0C
                    01  0192  4274          .BYTE   ^X01
                        0193  4275          VBUSDATA        <^X1E>,1                ;0D
                    01  0194  4276          .BYTE   ^X01
                        0195  4277          VBUSDATA        <^X1E>,1                ;0E
                    01  0196  4278          .BYTE   ^X01
                        0197  4279          VBUSDATA        <^X1E>,1                ;0F
                    01  0198  4280          .BYTE   ^X01
                        0199  4281          VBUSDATA        <^X1E>,1                ;10
                    01  019A  4282          .BYTE   ^X01
                        019B  4283          VBUSDATA        <^X1E>,1                ;11
                    01  019C  4284          .BYTE   ^X01
                        019D  4285          VBUSDATA        <^X1E>,1                ;12
                    01  019E  4286          .BYTE   ^X01
                        019F  4287          VBUSDATA        <^X1E>,1                ;13
                    01  01A0  4288          .BYTE   ^X01
                        01A1  4289          VBUSDATA        <^X1E>,1                ;14
                    01  01A2  4290          .BYTE   ^X01
                        01A3  4291          VBUSDATA        <^X1E>,1                ;15
                    01  01A4  4292          .BYTE   ^X01
                        01A5  4293          VBUSDATA        <^X1E>,1                ;16
                    01  01A6  4294          .BYTE   ^X01
                        01A7  4295          VBUSDATA        <^X1E>,1                ;17
                    01  01A8  4296          .BYTE   ^X01
                        01A9  4297          VBUSDATA        <^X1E>,1                ;18
                    01  01AA  4298          .BYTE   ^X01
                        01AB  4299          VBUSDATA        <^X1E>,1                ;19
                    01  01AC  4300          .BYTE   ^X01
                        01AD  4301          VBUSDATA        <^X1E>,1                ;1A
                    01  01AE  4302          .BYTE   ^X01
                        01AF  4303          VBUSDATA        <^X1E>,1                ;1B
                    01  01B0  4304          .BYTE   ^X01
                        01B1  4305          VBUSDATA        <^X1E>,1                ;1C
                    01  01B2  4306          .BYTE   ^X01
                        01B3  4307          VBUSDATA        <^X1E>,1                ;1D
                    01  01B4  4308          .BYTE   ^X01
                        01B5  4309          VBUSDATA        <^X1E>,1                ;1E
                    01  01B6  4310          .BYTE   ^X01
                        01B7  4311          VBUSDATA        <^X1E>,1                ;1F
                    01  01B8  4312          .BYTE   ^X01
```

```
                    01B9  4313          VBUSDATA        <^X1E>,1                ;20
                01  01BA  4314          .BYTE   ^X01
                    01BB  4315          VBUSDATA        <^X1E>,1                ;21
                01  01BC  4316          .BYTE   ^X01
                    01BD  4317          VBUSDATA        <^X1E>,1                ;22
                01  01BE  4318          .BYTE   ^X01
                    01BF  4319          VBUSDATA        <^X1E>,1                ;23
                01  01C0  4320          .BYTE   ^X01
                    01C1  4321          VBUSDATA        <^X1E>,1                ;24
                01  01C2  4322          .BYTE   ^X01
                    01C3  4323          VBUSDATA        <^X1E>,1                ;25
                01  01C4  4324          .BYTE   ^X01
                    01C5  4325          VBUSDATA        <^X1E>,1                ;26
                01  01C6  4326          .BYTE   ^X01
                    01C7  4327          VBUSDATA        <^X1E>,1                ;27
                01  01C8  4328          .BYTE   ^X01
                    01C9  4329          VBUSDATA        <^X1E>,1                ;28
                01  01CA  4330          .BYTE   ^X01
                    01CB  4331          VBUSDATA        <^X1E>,1                ;29
                01  01CC  4332          .BYTE   ^X01
                    01CD  4333          VBUSDATA        <^X1E>,1                ;2A
                01  01CE  4334          .BYTE   ^X01
                    01CF  4335          VBUSDATA        <^X1E>,1                ;2B
                01  01D0  4336          .BYTE   ^X01
                    01D1  4337          VBUSDATA        <^X1E>,1                ;2C
                01  01D2  4338          .BYTE   ^X01
                    01D3  4339          VBUSDATA        <^X1E>,1                ;2D
                01  01D4  4340          .BYTE   ^X01
                    01D5  4341          VBUSDATA        <^X1E>,1                ;2E
                01  01D6  4342          .BYTE   ^X01
                    01D7  4343          VBUSDATA        <^X1E>,1                ;2F
                01  01D8  4344          .BYTE   ^X01
                    01D9  4345          VBUSDATA        <^X1E>,1                ;30
                01  01DA  4346          .BYTE   ^X01
                    01DB  4347          VBUSDATA        <^X1E>,1                ;31
                01  01DC  4348          .BYTE   ^X01
                    01DD  4349          VBUSDATA        <^X1E>,1                ;32
                01  01DE  4350          .BYTE   ^X01
                    01DF  4351          VBUSDATA        <^X1E>,1                ;33
                01  01E0  4352          .BYTE   ^X01
                    01E1  4353          VBUSDATA        <^X1E>,1                ;34
                01  01E2  4354          .BYTE   ^X01
                    01E3  4355          VBUSDATA        <^X1E>,1                ;35
                01  01E4  4356          .BYTE   ^X01
                    01E5  4357          VBUSDATA        <^X1E>,1                ;36
                01  01E6  4358          .BYTE   ^X01
                    01E7  4359          VBUSDATA        <^X1E>,1                ;37
                01  01E8  4360          .BYTE   ^X01
                    01E9  4361          VBUSDATA        <^X1E>,1                ;38
                01  01EA  4362          .BYTE   ^X01
                    01EB  4363          VBUSDATA        <^X1E>,1                ;39
                01  01EC  4364          .BYTE   ^X01
                    01ED  4365          VBUSDATA        <^X1E>,1                ;3A
                01  01EE  4366          .BYTE   ^X01
                    01EF  4367          VBUSDATA        <^X1E>,1                ;3B
```

```
                    01  01F0  4368          .BYTE   ^X01
                        01F1  4369          VBUSDATA        <^X1E>,1                ;3C
                    01  01F2  4370          .BYTE   ^X01
                        01F3  4371          VBUSDATA        <^X1E>,1                ;3D
                    01  01F4  4372          .BYTE   ^X01
                        01F5  4373          VBUSDATA        <^X1E>,1                ;3E
                    01  01F6  4374          .BYTE   ^X01
                        01F7  4375          VBUSDATA        <^X1E>,1                ;3F
                    01  01F8  4376          .BYTE   ^X01
                        01F9  4377          VBUSDATA        <^X1E>,1                ;40
                    01  01FA  4378          .BYTE   ^X01
                        01FB  4379          VBUSDATA        <^X1E>,1                ;41
                    01  01FC  4380          .BYTE   ^X01
                        01FD  4381          VBUSDATA        <^X1E>,1                ;42
                    01  01FE  4382          .BYTE   ^X01
                        01FF  4383          VBUSDATA        <^X1E>,1                ;43
                    01  0200  4384          .BYTE   ^X01
                        0201  4385          VBUSDATA        <^X1E>,1                ;44
                    01  0202  4386          .BYTE   ^X01
                        0203  4387          VBUSDATA        <^X1E>,1                ;45
                    01  0204  4388          .BYTE   ^X01
                        0205  4389          VBUSDATA        <^X1E>,1                ;46
                    01  0206  4390          .BYTE   ^X01
                        0207  4391          VBUSDATA        <^X1E>,1                ;47
                    01  0208  4392          .BYTE   ^X01
                        0209  4393          VBUSDATA        <^X1E>,1                ;48
                    01  020A  4394          .BYTE   ^X01
                        020B  4395          VBUSDATA        <^X1E>,1                ;49
                    01  020C  4396          .BYTE   ^X01
                        020D  4397          VBUSDATA        <^X1E>,1                ;4A
                    01  020E  4398          .BYTE   ^X01
                        020F  4399          VBUSDATA        <^X1E>,1                ;4B
                    01  0210  4400          .BYTE   ^X01
                        0211  4401          VBUSDATA        <^X1E>,1                ;4C
                    01  0212  4402          .BYTE   ^X01
                        0213  4403          VBUSDATA        <^X1E>,1                ;4D
                    01  0214  4404          .BYTE   ^X01
                        0215  4405          VBUSDATA        <^X1E>,1                ;4E
                    01  0216  4406          .BYTE   ^X01
                        0217  4407          VBUSDATA        <^X1E>,1                ;4F
                    01  0218  4408          .BYTE   ^X01
                        0219  4409          VBUSDATA        <^X1E>,1                ;50
                    01  021A  4410          .BYTE   ^X01
                        021B  4411          VBUSDATA        <^X1E>,1                ;51
                    01  021C  4412          .BYTE   ^X01
                        021D  4413          VBUSDATA        <^X1E>,1                ;52
                    01  021E  4414          .BYTE   ^X01
                        021F  4415          VBUSDATA        <^X1E>,1                ;53
                    01  0220  4416          .BYTE   ^X01
                        0221  4417          VBUSDATA        <^X1E>,1                ;54
                    01  0222  4418          .BYTE   ^X01
                        0223  4419          VBUSDATA        <^X1E>,1                ;55
                    01  0224  4420          .BYTE   ^X01
                        0225  4421          VBUSDATA        <^X1E>,1                ;56
                    01  0226  4422          .BYTE   ^X01
```

```
                0227  4423          VBUSDATA        <^X1E>,1                ;57
          01    0228  4424          .BYTE   ^X01
                0229  4425          VBUSDATA        <^X1E>,1                ;58
          01    022A  4426          .BYTE   ^X01
                022B  4427          VBUSDATA        <^X1E>,1                ;59
          01    022C  4428          .BYTE   ^X01
                022D  4429          VBUSDATA        <^X1E>,1                ;5A
          01    022E  4430          .BYTE   ^X01
                022F  4431          VBUSDATA        <^X1E>,1                ;5B
          01    0230  4432          .BYTE   ^X01
                0231  4433          VBUSDATA        <^X1E>,1                ;5C
          01    0232  4434          .BYTE   ^X01
                0233  4435          VBUSDATA        <^X1E>,1                ;5D
          01    0234  4436          .BYTE   ^X01
                0235  4437          VBUSDATA        <^X1E>,1                ;5E
          01    0236  4438          .BYTE   ^X01
                0237  4439          VBUSDATA        <^X1E>,1                ;5F
          01    0238  4440          .BYTE   ^X01
                0239  4441          VBUSDATA        <^X1E>,1                ;60
          01    023A  4442          .BYTE   ^X01
                023B  4443          VBUSDATA        <^X1E>,1                ;61
          01    023C  4444          .BYTE   ^X01
                023D  4445          VBUSDATA        <^X1E>,1                ;62
          01    023E  4446          .BYTE   ^X01
                023F  4447          VBUSDATA        <^X1E>,1                ;63
          01    0240  4448          .BYTE   ^X01
                0241  4449          VBUSDATA        <^X1E>,1                ;64
          01    0242  4450          .BYTE   ^X01
                0243  4451          VBUSDATA        <^X1E>,1                ;65
          01    0244  4452          .BYTE   ^X01
                0245  4453          VBUSDATA        <^X1E>,1                ;66
          01    0246  4454          .BYTE   ^X01
                0247  4455          VBUSDATA        <^X1E>,1                ;67
          01    0248  4456          .BYTE   ^X01
                0249  4457          VBUSDATA        <^X1E>,1                ;68
          01    024A  4458          .BYTE   ^X01
                024B  4459          VBUSDATA        <^X1E>,1                ;69
          01    024C  4460          .BYTE   ^X01
                024D  4461          VBUSDATA        <^X1E>,1                ;6A
          01    024E  4462          .BYTE   ^X01
                024F  4463          VBUSDATA        <^X1E>,1                ;6B
          01    0250  4464          .BYTE   ^X01
                0251  4465          VBUSDATA        <^X1E>,1                ;6C
          01    0252  4466          .BYTE   ^X01
                0253  4467          VBUSDATA        <^X1E>,1                ;6D
          01    0254  4468          .BYTE   ^X01
                0255  4469          VBUSDATA        <^X1E>,1                ;6E
          01    0256  4470          .BYTE   ^X01
                0257  4471          VBUSDATA        <^X1E>,1                ;6F
          01    0258  4472          .BYTE   ^X01
                0259  4473          VBUSDATA        <^X1E>,1                ;70
          01    025A  4474          .BYTE   ^X01
                025B  4475          VBUSDATA        <^X1E>,1                ;71
          01    025C  4476          .BYTE   ^X01
                025D  4477          VBUSDATA        <^X1E>,1                ;72
```

```
01  025E  4478          .BYTE   ^X01
    025F  4479          VBUSDATA        <^X1E>,1                ;73
01  0260  4480          .BYTE   ^X01
    0261  4481          VBUSDATA        <^X1E>,1                ;74
01  0262  4482          .BYTE   ^X01
    0263  4483          VBUSDATA        <^X1E>,1                ;75
01  0264  4484          .BYTE   ^X01
    0265  4485          VBUSDATA        <^X1E>,1                ;76
01  0266  4486          .BYTE   ^X01
    0267  4487          VBUSDATA        <^X1E>,1                ;77
01  0268  4488          .BYTE   ^X01
    0269  4489          VBUSDATA        <^X1E>,1                ;78
01  026A  4490          .BYTE   ^X01
    026B  4491          VBUSDATA        <^X1E>,1                ;79
01  026C  4492          .BYTE   ^X01
    026D  4493          VBUSDATA        <^X1E>,1                ;7A
01  026E  4494          .BYTE   ^X01
    026F  4495          VBUSDATA        <^X1E>,1                ;7B
01  0270  4496          .BYTE   ^X01
    0271  4497          VBUSDATA        <^X1E>,1                ;7C
01  0272  4498          .BYTE   ^X01
    0273  4499          VBUSDATA        <^X1E>,1                ;7D
01  0274  4500          .BYTE   ^X01
    0275  4501          VBUSDATA        <^X1E>,1                ;7E
01  0276  4502          .BYTE   ^X01
    0277  4503          VBUSDATA        <^X1E>,1                ;7F
01  0278  4504          .BYTE   ^X01
    0279  4505          VBUSDATA        <^X1E>,1                ;80
01  027A  4506          .BYTE   ^X01
    027B  4507          VBUSDATA        <^X1E>,1                ;81
01  027C  4508          .BYTE   ^X01
    027D  4509          VBUSDATA        <^X1E>,1                ;82
01  027E  4510          .BYTE   ^X01
    027F  4511          VBUSDATA        <^X1E>,1                ;83
01  0280  4512          .BYTE   ^X01
    0281  4513          VBUSDATA        <^X1E>,1                ;84
01  0282  4514          .BYTE   ^X01
    0283  4515          VBUSDATA        <^X1E>,1                ;85
01  0284  4516          .BYTE   ^X01
    0285  4517          VBUSDATA        <^X1E>,1                ;86
01  0286  4518          .BYTE   ^X01
    0287  4519          VBUSDATA        <^X1E>,1                ;87
01  0288  4520          .BYTE   ^X01
    0289  4521          VBUSDATA        <^X1E>,1                ;88
01  028A  4522          .BYTE   ^X01
    028B  4523          VBUSDATA        <^X1E>,1                ;89
01  028C  4524          .BYTE   ^X01
    028D  4525          VBUSDATA        <^X1E>,1                ;8A
01  028E  4526          .BYTE   ^X01
    028F  4527          VBUSDATA        <^X1E>,1                ;8B
01  0290  4528          .BYTE   ^X01
    0291  4529          VBUSDATA        <^X1E>,1                ;8C
01  0292  4530          .BYTE   ^X01
    0293  4531          VBUSDATA        <^X1E>,1                ;8D
01  0294  4532          .BYTE   ^X01
```

```
                0295  4533          VBUSDATA        <^X1E>,1                ;8E
01              0296  4534          .BYTE   ^X01
                0297  4535          VBUSDATA        <^X1E>,1                ;8F
01              0298  4536          .BYTE   ^X01
                0299  4537          VBUSDATA        <^X1E>,1                ;90
01              029A  4538          .BYTE   ^X01
                029B  4539          VBUSDATA        <^X1E>,1                ;91
01              029C  4540          .BYTE   ^X01
                029D  4541          VBUSDATA        <^X1E>,1                ;92
01              029E  4542          .BYTE   ^X01
                029F  4543          VBUSDATA        <^X1E>,1                ;93
01              02A0  4544          .BYTE   ^X01
                02A1  4545          VBUSDATA        <^X1E>,1                ;94
01              02A2  4546          .BYTE   ^X01
                02A3  4547          VBUSDATA        <^X1E>,1                ;95
01              02A4  4548          .BYTE   ^X01
                02A5  4549          VBUSDATA        <^X1E>,1                ;96
01              02A6  4550          .BYTE   ^X01
                02A7  4551          VBUSDATA        <^X1E>,1                ;97
01              02A8  4552          .BYTE   ^X01
                02A9  4553          VBUSDATA        <^X1E>,1                ;98
01              02AA  4554          .BYTE   ^X01
                02AB  4555          VBUSDATA        <^X1E>,1                ;99
01              02AC  4556          .BYTE   ^X01
                02AD  4557          VBUSDATA        <^X1E>,1                ;9A
01              02AE  4558          .BYTE   ^X01
                02AF  4559          VBUSDATA        <^X1E>,1                ;9B
01              02B0  4560          .BYTE   ^X01
                02B1  4561          VBUSDATA        <^X1E>,1                ;9C
01              02B2  4562          .BYTE   ^X01
                02B3  4563          VBUSDATA        <^X1E>,1                ;9D
01              02B4  4564          .BYTE   ^X01
                02B5  4565          VBUSDATA        <^X1E>,1                ;9E
01              02B6  4566          .BYTE   ^X01
                02B7  4567          VBUSDATA        <^X1E>,1                ;9F
01              02B8  4568          .BYTE   ^X01
                02B9  4569          VBUSDATA        <^X1E>,1                ;A0
01              02BA  4570          .BYTE   ^X01
                02BB  4571          VBUSDATA        <^X1E>,1                ;A1
01              02BC  4572          .BYTE   ^X01
                02BD  4573          VBUSDATA        <^X1E>,1                ;A2
01              02BE  4574          .BYTE   ^X01
                02BF  4575          VBUSDATA        <^X1E>,1                ;A3
01              02C0  4576          .BYTE   ^X01
                02C1  4577          VBUSDATA        <^X1E>,1                ;A4
01              02C2  4578          .BYTE   ^X01
                02C3  4579          VBUSDATA        <^X1E>,1                ;A5
01              02C4  4580          .BYTE   ^X01
                02C5  4581          VBUSDATA        <^X1E>,1                ;A6
01              02C6  4582          .BYTE   ^X01
                02C7  4583          VBUSDATA        <^X1E>,1                ;A7
01              02C8  4584          .BYTE   ^X01
                02C9  4585          VBUSDATA        <^X1E>,1                ;A8
01              02CA  4586          .BYTE   ^X01
                02CB  4587          VBUSDATA        <^X1E>,1                ;A9
```

```
                01  02CC  4588          .BYTE   ^X01
                    02CD  4589          VBUSDATA        <^X1E>,1                ;AA
                01  02CE  4590          .BYTE   ^X01
                    02CF  4591          VBUSDATA        <^X1E>,1                ;AB
                01  02D0  4592          .BYTE   ^X01
                    02D1  4593          VBUSDATA        <^X1E>,1                ;AC
                01  02D2  4594          .BYTE   ^X01
                    02D3  4595          VBUSDATA        <^X1E>,1                ;AD
                01  02D4  4596          .BYTE   ^X01
                    02D5  4597          VBUSDATA        <^X1E>,1                ;AE
                01  02D6  4598          .BYTE   ^X01
                    02D7  4599          VBUSDATA        <^X1E>,1                ;AF
                01  02D8  4600          .BYTE   ^X01
                    02D9  4601          VBUSDATA        <^X1E>,1                ;B0
                01  02DA  4602          .BYTE   ^X01
                    02DB  4603          VBUSDATA        <^X1E>,1                ;B1
                01  02DC  4604          .BYTE   ^X01
                    02DD  4605          VBUSDATA        <^X1E>,1                ;B2
                01  02DE  4606          .BYTE   ^X01
                    02DF  4607          VBUSDATA        <^X1E>,1                ;B3
                01  02E0  4608          .BYTE   ^X01
                    02E1  4609          VBUSDATA        <^X1E>,1                ;B4
                01  02E2  4610          .BYTE   ^X01
                    02E3  4611          VBUSDATA        <^X1E>,1                ;B5
                01  02E4  4612          .BYTE   ^X01
                    02E5  4613          VBUSDATA        <^X1E>,1                ;B6
                01  02E6  4614          .BYTE   ^X01
                    02E7  4615          VBUSDATA        <^X1E>,1                ;B7
                01  02E8  4616          .BYTE   ^X01
                    02E9  4617          VBUSDATA        <^X1E>,1                ;B8
                01  02EA  4618          .BYTE   ^X01
                    02EB  4619          VBUSDATA        <^X1E>,1                ;B9
                01  02EC  4620          .BYTE   ^X01
                    02ED  4621          VBUSDATA        <^X1E>,1                ;BA
                01  02EE  4622          .BYTE   ^X01
                    02EF  4623          VBUSDATA        <^X1E>,1                ;BB
                01  02F0  4624          .BYTE   ^X01
                    02F1  4625          VBUSDATA        <^X1E>,1                ;BC
                01  02F2  4626          .BYTE   ^X01
                    02F3  4627          VBUSDATA        <^X1E>,1                ;BD
                01  02F4  4628          .BYTE   ^X01
                    02F5  4629          VBUSDATA        <^X1E>,1                ;BE
                01  02F6  4630          .BYTE   ^X01
                    02F7  4631          VBUSDATA        <^X1E>,1                ;BF
                01  02F8  4632          .BYTE   ^X01
                    02F9  4633          VBUSDATA        <^X1E>,1                ;C0
                01  02FA  4634          .BYTE   ^X01
                    02FB  4635          VBUSDATA        <^X1E>,1                ;C1
                01  02FC  4636          .BYTE   ^X01
                    02FD  4637          VBUSDATA        <^X1E>,1                ;C2
                01  02FE  4638          .BYTE   ^X01
                    02FF  4639          VBUSDATA        <^X1E>,1                ;C3
                01  0300  4640          .BYTE   ^X01
                    0301  4641          VBUSDATA        <^X1E>,1                ;C4
                01  0302  4642          .BYTE   ^X01
```

```
                    0303  4643          VBUSDATA        <^X1E>,1                ;C5
01                  0304  4644          .BYTE   ^X01
                    0305  4645          VBUSDATA        <^X1E>,1                ;C6
01                  0306  4646          .BYTE   ^X01
                    0307  4647          VBUSDATA        <^X1E>,1                ;C7
01                  0308  4648          .BYTE   ^X01
                    0309  4649          VBUSDATA        <^X1E>,1                ;C8
01                  030A  4650          .BYTE   ^X01
                    030B  4651          VBUSDATA        <^X1E>,1                ;C9
01                  030C  4652          .BYTE   ^X01
                    030D  4653          VBUSDATA        <^X1E>,1                ;CA
01                  030E  4654          .BYTE   ^X01
                    030F  4655          VBUSDATA        <^X1E>,1                ;CB
01                  0310  4656          .BYTE   ^X01
                    0311  4657          VBUSDATA        <^X1E>,1                ;CC
01                  0312  4658          .BYTE   ^X01
                    0313  4659          VBUSDATA        <^X1E>,1                ;CD
01                  0314  4660          .BYTE   ^X01
                    0315  4661          VBUSDATA        <^X1E>,1                ;CE
01                  0316  4662          .BYTE   ^X01
                    0317  4663          VBUSDATA        <^X1E>,1                ;CF
01                  0318  4664          .BYTE   ^X01
                    0319  4665          VBUSDATA        <^X1E>,1                ;D0
01                  031A  4666          .BYTE   ^X01
                    031B  4667          VBUSDATA        <^X1E>,1                ;D1
01                  031C  4668          .BYTE   ^X01
                    031D  4669          VBUSDATA        <^X1E>,1                ;D2
01                  031E  4670          .BYTE   ^X01
                    031F  4671          VBUSDATA        <^X1E>,1                ;D3
01                  0320  4672          .BYTE   ^X01
                    0321  4673          VBUSDATA        <^X1E>,1                ;D4
01                  0322  4674          .BYTE   ^X01
                    0323  4675          VBUSDATA        <^X1E>,1                ;D5
01                  0324  4676          .BYTE   ^X01
                    0325  4677          VBUSDATA        <^X1E>,1                ;D6
01                  0326  4678          .BYTE   ^X01
                    0327  4679          VBUSDATA        <^X1E>,1                ;D7
01                  0328  4680          .BYTE   ^X01
                    0329  4681          VBUSDATA        <^X1E>,1                ;D8
01                  032A  4682          .BYTE   ^X01
                    032B  4683          VBUSDATA        <^X1E>,1                ;D9
01                  032C  4684          .BYTE   ^X01
                    032D  4685          VBUSDATA        <^X1E>,1                ;DA
01                  032E  4686          .BYTE   ^X01
                    032F  4687          VBUSDATA        <^X1E>,1                ;DB
01                  0330  4688          .BYTE   ^X01
                    0331  4689          VBUSDATA        <^X1E>,1                ;DC
01                  0332  4690          .BYTE   ^X01
                    0333  4691          VBUSDATA        <^X1E>,1                ;DD
01                  0334  4692          .BYTE   ^X01
                    0335  4693          VBUSDATA        <^X1E>,1                ;DE
01                  0336  4694          .BYTE   ^X01
                    0337  4695          VBUSDATA        <^X1E>,1                ;DF
01                  0338  4696          .BYTE   ^X01
                    0339  4697          VBUSDATA        <^X1E>,1                ;E0
```

```
01  033A  4698          .BYTE   ^X01
    033B  4699          VBUSDATA        <^X1E>,1                ;E1
01  033C  4700          .BYTE   ^X01
    033D  4701          VBUSDATA        <^X1E>,1                ;E2
01  033E  4702          .BYTE   ^X01
    033F  4703          VBUSDATA        <^X1E>,1                ;E3
01  0340  4704          .BYTE   ^X01
    0341  4705          VBUSDATA        <^X1E>,1                ;E4
01  0342  4706          .BYTE   ^X01
    0343  4707          VBUSDATA        <^X1E>,1                ;E5
01  0344  4708          .BYTE   ^X01
    0345  4709          VBUSDATA        <^X1E>,1                ;E6
01  0346  4710          .BYTE   ^X01
    0347  4711          VBUSDATA        <^X1E>,1                ;E7
01  0348  4712          .BYTE   ^X01
    0349  4713          VBUSDATA        <^X1E>,1                ;E8
01  034A  4714          .BYTE   ^X01
    034B  4715          VBUSDATA        <^X1E>,1                ;E9
01  034C  4716          .BYTE   ^X01
    034D  4717          VBUSDATA        <^X1E>,1                ;EA
01  034E  4718          .BYTE   ^X01
    034F  4719          VBUSDATA        <^X1E>,1                ;EB
01  0350  4720          .BYTE   ^X01
    0351  4721          VBUSDATA        <^X1E>,1                ;EC
01  0352  4722          .BYTE   ^X01
    0353  4723          VBUSDATA        <^X1E>,1                ;ED
01  0354  4724          .BYTE   ^X01
    0355  4725          VBUSDATA        <^X1E>,1                ;EE
01  0356  4726          .BYTE   ^X01
    0357  4727          VBUSDATA        <^X1E>,1                ;EF
01  0358  4728          .BYTE   ^X01
    0359  4729          VBUSDATA        <^X1E>,1                ;F0
01  035A  4730          .BYTE   ^X01
    035B  4731          VBUSDATA        <^X1E>,1                ;F1
01  035C  4732          .BYTE   ^X01
    035D  4733          VBUSDATA        <^X1E>,1                ;F2
01  035E  4734          .BYTE   ^X01
    035F  4735          VBUSDATA        <^X1E>,1                ;F3
01  0360  4736          .BYTE   ^X01
    0361  4737          VBUSDATA        <^X1E>,1                ;F4
01  0362  4738          .BYTE   ^X01
    0363  4739          VBUSDATA        <^X1E>,1                ;F5
01  0364  4740          .BYTE   ^X01
    0365  4741          VBUSDATA        <^X1E>,1                ;F6
01  0366  4742          .BYTE   ^X01
    0367  4743          VBUSDATA        <^X1E>,1                ;F7
01  0368  4744          .BYTE   ^X01
    0369  4745          VBUSDATA        <^X1E>,1                ;F8
01  036A  4746          .BYTE   ^X01
    036B  4747          VBUSDATA        <^X1E>,1                ;F9
01  036C  4748          .BYTE   ^X01
    036D  4749          VBUSDATA        <^X1E>,1                ;FA
01  036E  4750          .BYTE   ^X01
    036F  4751          VBUSDATA        <^X1E>,1                ;FB
01  0370  4752          .BYTE   ^X01
```

```
                    0371  4753          VBUSDATA        <^X1E>,1                ;FC
01                  0372  4754          .BYTE   ^X01
                    0373  4755          VBUSDATA        <^X1E>,1                ;FD
01                  0374  4756          .BYTE   ^X01
                    0375  4757          VBUSDATA        <^X1E>,1                ;FE
01                  0376  4758          .BYTE   ^X01
                    0377  4759          VBUSDATA        <^X1E>,1                ;FF
                    0378  4760
                    0378  4761 ;
                    0378  4762 ; Expected VBUS bit values for Subtest 3
                    0378  4763 ;
                    0378  4764 ; The following is a table of all the VBUS bit values for an IRD1 with the FPA
                    0378  4765 ; enabled. The hex numbers in the comments to the right indicate the range of
                    0378  4766 ; loop counter values for each row of address words.
                    0378  4767 ;
                    0378  4768
01                  0378  4769 11$:    .BYTE   ^X01
                    0379  4770          VBUSDATA        <^X1E>,1                ;00
01                  037A  4771          .BYTE   ^X01
                    037B  4772          VBUSDATA        <^X1E>,1                ;01
01                  037C  4773          .BYTE   ^X01
                    037D  4774          VBUSDATA        <^X1E>,1                ;02
01                  037E  4775          .BYTE   ^X01
                    037F  4776          VBUSDATA        <^X1E>,1                ;03
01                  0380  4777          .BYTE   ^X01
                    0381  4778          VBUSDATA        <^X1E>,1                ;04
01                  0382  4779          .BYTE   ^X01
                    0383  4780          VBUSDATA        <^X1E>,1                ;05
01                  0384  4781          .BYTE   ^X01
                    0385  4782          VBUSDATA        <^X1E>,1                ;06
01                  0386  4783          .BYTE   ^X01
                    0387  4784          VBUSDATA        <^X1E>,1                ;07
01                  0388  4785          .BYTE   ^X01
                    0389  4786          VBUSDATA        <^X1E>,1                ;08
01                  038A  4787          .BYTE   ^X01
                    038B  4788          VBUSDATA        <^X1E>,1                ;09
01                  038C  4789          .BYTE   ^X01
                    038D  4790          VBUSDATA        <^X1E>,1                ;0A
01                  038E  4791          .BYTE   ^X01
                    038F  4792          VBUSDATA        <^X1E>,1                ;0B
01                  0390  4793          .BYTE   ^X01
                    0391  4794          VBUSDATA        <^X1E>,1                ;0C
01                  0392  4795          .BYTE   ^X01
                    0393  4796          VBUSDATA        <^X1E>,1                ;0D
01                  0394  4797          .BYTE   ^X01
                    0395  4798          VBUSDATA        <^X1E>,1                ;0E
01                  0396  4799          .BYTE   ^X01
                    0397  4800          VBUSDATA        <^X1E>,1                ;0F
01                  0398  4801          .BYTE   ^X01
                    0399  4802          VBUSDATA        <^X1E>,1                ;10
01                  039A  4803          .BYTE   ^X01
                    039B  4804          VBUSDATA        <^X1E>,1                ;11
01                  039C  4805          .BYTE   ^X01
                    039D  4806          VBUSDATA        <^X1E>,1                ;12
01                  039E  4807          .BYTE   ^X01
```

```
                039F  4808          VBUSDATA        <^X1E>,1                ;13
            01  03A0  4809          .BYTE   ^X01
                03A1  4810          VBUSDATA        <^X1E>,1                ;14
            01  03A2  4811          .BYTE   ^X01
                03A3  4812          VBUSDATA        <^X1E>,1                ;15
            01  03A4  4813          .BYTE   ^X01
                03A5  4814          VBUSDATA        <^X1E>,1                ;16
            01  03A6  4815          .BYTE   ^X01
                03A7  4816          VBUSDATA        <^X1E>,1                ;17
            01  03A8  4817          .BYTE   ^X01
                03A9  4818          VBUSDATA        <^X1E>,1                ;18
            01  03AA  4819          .BYTE   ^X01
                03AB  4820          VBUSDATA        <^X1E>,1                ;19
            01  03AC  4821          .BYTE   ^X01
                03AD  4822          VBUSDATA        <^X1E>,1                ;1A
            01  03AE  4823          .BYTE   ^X01
                03AF  4824          VBUSDATA        <^X1E>,1                ;1B
            01  03B0  4825          .BYTE   ^X01
                03B1  4826          VBUSDATA        <^X1E>,1                ;1C
            01  03B2  4827          .BYTE   ^X01
                03B3  4828          VBUSDATA        <^X1E>,1                ;1D
            01  03B4  4829          .BYTE   ^X01
                03B5  4830          VBUSDATA        <^X1E>,1                ;1E
            01  03B6  4831          .BYTE   ^X01
                03B7  4832          VBUSDATA        <^X1E>,1                ;1F
            01  03B8  4833          .BYTE   ^X01
                03B9  4834          VBUSDATA        <^X1E>,1                ;20
            01  03BA  4835          .BYTE   ^X01
                03BB  4836          VBUSDATA        <^X1E>,1                ;21
            01  03BC  4837          .BYTE   ^X01
                03BD  4838          VBUSDATA        <^X1E>,1                ;22
            01  03BE  4839          .BYTE   ^X01
                03BF  4840          VBUSDATA        <^X1E>,1                ;23
            01  03C0  4841          .BYTE   ^X01
                03C1  4842          VBUSDATA        <^X1E>,1                ;24
            01  03C2  4843          .BYTE   ^X01
                03C3  4844          VBUSDATA        <^X1E>,1                ;25
            01  03C4  4845          .BYTE   ^X01
                03C5  4846          VBUSDATA        <^X1E>,1                ;26
            01  03C6  4847          .BYTE   ^X01
                03C7  4848          VBUSDATA        <^X1E>,1                ;27
            01  03C8  4849          .BYTE   ^X01
                03C9  4850          VBUSDATA        <^X1E>,1                ;28
            01  03CA  4851          .BYTE   ^X01
                03CB  4852          VBUSDATA        <^X1E>,1                ;29
            01  03CC  4853          .BYTE   ^X01
                03CD  4854          VBUSDATA        <^X1E>,1                ;2A
            01  03CE  4855          .BYTE   ^X01
                03CF  4856          VBUSDATA        <^X1E>,1                ;2B
            01  03D0  4857          .BYTE   ^X01
                03D1  4858          VBUSDATA        <^X1E>,1                ;2C
            01  03D2  4859          .BYTE   ^X01
                03D3  4860          VBUSDATA        <^X1E>,1                ;2D
            01  03D4  4861          .BYTE   ^X01
                03D5  4862          VBUSDATA        <^X1E>,1                ;2E
```

```
01  03D6  4863          .BYTE   ^X01
    03D7  4864          VBUSDATA        <^X1E>,1                ;2F
01  03D8  4865          .BYTE   ^X01
    03D9  4866          VBUSDATA        <^X1E>,1                ;30
01  03DA  4867          .BYTE   ^X01
    03DB  4868          VBUSDATA        <^X1E>,1                ;31
01  03DC  4869          .BYTE   ^X01
    03DD  4870          VBUSDATA        <^X1E>,1                ;32
01  03DE  4871          .BYTE   ^X01
    03DF  4872          VBUSDATA        <^X1E>,1                ;33
01  03E0  4873          .BYTE   ^X01
    03E1  4874          VBUSDATA        <^X1E>,1                ;34
01  03E2  4875          .BYTE   ^X01
    03E3  4876          VBUSDATA        <^X1E>,1                ;35
01  03E4  4877          .BYTE   ^X01
    03E5  4878          VBUSDATA        <^X1E>,1                ;36
01  03E6  4879          .BYTE   ^X01
    03E7  4880          VBUSDATA        <^X1E>,1                ;37
01  03E8  4881          .BYTE   ^X01
    03E9  4882          VBUSDATA        <^X1E>,1                ;38
01  03EA  4883          .BYTE   ^X01
    03EB  4884          VBUSDATA        <^X1E>,1                ;39
01  03EC  4885          .BYTE   ^X01
    03ED  4886          VBUSDATA        <^X1E>,1                ;3A
01  03EE  4887          .BYTE   ^X01
    03EF  4888          VBUSDATA        <^X1E>,1                ;3B
01  03F0  4889          .BYTE   ^X01
    03F1  4890          VBUSDATA        <^X1E>,1                ;3C
01  03F2  4891          .BYTE   ^X01
    03F3  4892          VBUSDATA        <^X1E>,1                ;3D
01  03F4  4893          .BYTE   ^X01
    03F5  4894          VBUSDATA        <^X1E>,1                ;3E
01  03F6  4895          .BYTE   ^X01
    03F7  4896          VBUSDATA        <^X1E>,1                ;3F
01  03F8  4897          .BYTE   ^X01
    03F9  4898          VBUSDATA        <^X1E>,1                ;40
01  03FA  4899          .BYTE   ^X01
    03FB  4900          VBUSDATA        <^X1E>,1                ;41
01  03FC  4901          .BYTE   ^X01
    03FD  4902          VBUSDATA        <^X1E>,1                ;42
01  03FE  4903          .BYTE   ^X01
    03FF  4904          VBUSDATA        <^X1E>,1                ;43
01  0400  4905          .BYTE   ^X01
    0401  4906          VBUSDATA        <^X1E>,1                ;44
01  0402  4907          .BYTE   ^X01
    0403  4908          VBUSDATA        <^X1E>,1                ;45
01  0404  4909          .BYTE   ^X01
    0405  4910          VBUSDATA        <^X1E>,1                ;46
01  0406  4911          .BYTE   ^X01
    0407  4912          VBUSDATA        <^X1E>,'                ;47
01  0408  4913          .BYTE   ^X01
    0409  4914          VBUSDATA        <^X1E>,1                ;48
01  040A  4915          .BYTE   ^X01
    040B  4916          VBUSDATA        <^X1E>,1                ;49
01  040C  4917          .BYTE   ^X01
```

```
                040D  4918          VBUSDATA        <^X1E>,1                ;4A
            01  040E  4919          .BYTE   ^X01
                040F  4920          VBUSDATA        <^X1E>,1                ;4B
            01  0410  4921          .BYTE   ^X01
                0411  4922          VBUSDATA        <^X1E>,1                ;4C
            01  0412  4923          .BYTE   ^X01
                0413  4924          VBUSDATA        <^X1E>,1                ;4D
            01  0414  4925          .BYTE   ^X01
                0415  4926          VBUSDATA        <^X1E>,1                ;4E
            01  0416  4927          .BYTE   ^X01
                0417  4928          VBUSDATA        <^X1E>,1                ;4F
            01  0418  4929          .BYTE   ^X01
                0419  4930          VBUSDATA        <^X1E>,1                ;50
            01  041A  4931          .BYTE   ^X01
                041B  4932          VBUSDATA        <^X1E>,1                ;51
            01  041C  4933          .BYTE   ^X01
                041D  4934          VBUSDATA        <^X1E>,1                ;52
            01  041E  4935          .BYTE   ^X01
                041F  4936          VBUSDATA        <^X1E>,1                ;53
            01  0420  4937          .BYTE   ^X01
                0421  4938          VBUSDATA        <^X1E>,1                ;54
            01  0422  4939          .BYTE   ^X01
                0423  4940          VBUSDATA        <^X1E>,1                ;55
            01  0424  4941          .BYTE   ^X01
                0425  4942          VBUSDATA        <^X1E>,1                ;56
            01  0426  4943          .BYTE   ^X01
                0427  4944          VBUSDATA        <^X1E>,1                ;57
            01  0428  4945          .BYTE   ^X01
                0429  4946          VBUSDATA        <^X1E>,1                ;58
            01  042A  4947          .BYTE   ^X01
                042B  4948          VBUSDATA        <^X1E>,1                ;59
            01  042C  4949          .BYTE   ^X01
                042D  4950          VBUSDATA        <^X1E>,1                ;5A
            01  042E  4951          .BYTE   ^X01
                042F  4952          VBUSDATA        <^X1E>,1                ;5B
            01  0430  4953          .BYTE   ^X01
                0431  4954          VBUSDATA        <^X1E>,1                ;5C
            01  0432  4955          .BYTE   ^X01
                0433  4956          VBUSDATA        <^X1E>,1                ;5D
            01  0434  4957          .BYTE   ^X01
                0435  4958          VBUSDATA        <^X1E>,1                ;5E
            01  0436  4959          .BYTE   ^X01
                0437  4960          VBUSDATA        <^X1E>,1                ;5F
            01  0438  4961          .BYTE   ^X01
                0439  4962          VBUSDATA        <^X1E>,1                ;60
            01  043A  4963          .BYTE   ^X01
                043B  4964          VBUSDATA        <^X1E>,1                ;61
            01  043C  4965          .BYTE   ^X01
                043D  4966          VBUSDATA        <^X1E>,1                ;62
            01  043E  4967          .BYTE   ^X01
                043F  4968          VBUSDATA        <^X1E>,1                ;63
            01  0440  4969          .BYTE   ^X01
                0441  4970          VBUSDATA        <^X1E>,1                ;64
            01  0442  4971          .BYTE   ^X01
                0443  4972          VBUSDATA        <^X1E>,1                ;65
```

```
01  0444  4973          .BYTE   ^X01
      0445  4974          VBUSDATA        <^X1E>,1                ;66
01  0446  4975          .BYTE   ^X01
      0447  4976          VBUSDATA        <^X1E>,1                ;67
01  0448  4977          .BYTE   ^X01
      0449  4978          VBUSDATA        <^X1E>,1                ;68
01  044A  4979          .BYTE   ^X01
      044B  4980          VBUSDATA        <^X1E>,1                ;69
01  044C  4981          .BYTE   ^X01
      044D  4982          VBUSDATA        <^X1E>,1                ;6A
01  044E  4983          .BYTE   ^X01
      044F  4984          VBUSDATA        <^X1E>,1                ;6B
01  0450  4985          .BYTE   ^X01
      0451  4986          VBUSDATA        <^X1E>,1                ;6C
01  0452  4987          .BYTE   ^X01
      0453  4988          VBUSDATA        <^X1E>,1                ;6D
01  0454  4989          .BYTE   ^X01
      0455  4990          VBUSDATA        <^X1E>,1                ;6E
01  0456  4991          .BYTE   ^X01
      0457  4992          VBUSDATA        <^X1E>,1                ;6F
01  0458  4993          .BYTE   ^X01
      0459  4994          VBUSDATA        <^X1E>,1                ;70
01  045A  4995          .BYTE   ^X01
      045B  4996          VBUSDATA        <^X1E>,1                ;71
01  045C  4997          .BYTE   ^X01
      045D  4998          VBUSDATA        <^X1E>,1                ;72
01  045E  4999          .BYTE   ^X01
      045F  5000          VBUSDATA        <^X1E>,1                ;73
01  0460  5001          .BYTE   ^X01
      0461  5002          VBUSDATA        <^X1E>,1                ;74
01  0462  5003          .BYTE   ^X01
      0463  5004          VBUSDATA        <^X1E>,1                ;75
01  0464  5005          .BYTE   ^X01
      0465  5006          VBUSDATA        <^X1E>,1                ;76
01  0466  5007          .BYTE   ^X01
      0467  5008          VBUSDATA        <^X1E>,1                ;77
01  0468  5009          .BYTE   ^X01
      0469  5010          VBUSDATA        <^X1E>,1                ;78
01  046A  5011          .BYTE   ^X01
      046B  5012          VBUSDATA        <^X1E>,1                ;79
01  046C  5013          .BYTE   ^X01
      046D  5014          VBUSDATA        <^X1E>,1                ;7A
01  046E  5015          .BYTE   ^X01
      046F  5016          VBUSDATA        <^X1E>,1                ;7B
01  0470  5017          .BYTE   ^X01
      0471  5018          VBUSDATA        <^X1E>,1                ;7C
01  0472  5019          .BYTE   ^X01
      0473  5020          VBUSDATA        <^X1E>,1                ;7D
01  0474  5021          .BYTE   ^X01
      0475  5022          VBUSDATA        <^X1E>,1                ;7E
01  0476  5023          .BYTE   ^X01
      0477  5024          VBUSDATA        <^X1E>,1                ;7F
01  0478  5025          .BYTE   ^X01
      0479  5026          VBUSDATA        <^X1E>,1                ;80
01  047A  5027          .BYTE   ^X01
```

```
        047B  5028        VBUSDATA        <^X1E>,1                ;81
   01   047C  5029        .BYTE   ^X01
        047D  5030        VBUSDATA        <^X1E>,1                ;82
   01   047E  5031        .BYTE   ^X01
        047F  5032        VBUSDATA        <^X1E>,1                ;83
   01   0480  5033        .BYTE   ^X01
        0481  5034        VBUSDATA        <^X1E>,1                ;84
   01   0482  5035        .BYTE   ^X01
        0483  5036        VBUSDATA        <^X1E>,1                ;85
   01   0484  5037        .BYTE   ^X01
        0485  5038        VBUSDATA        <^X1E>,1                ;86
   01   0486  5039        .BYTE   ^X01
        0487  5040        VBUSDATA        <^X1E>,1                ;87
   01   0488  5041        .BYTE   ^X01
        0489  5042        VBUSDATA        <^X1E>,1                ;88
   01   048A  5043        .BYTE   ^X01
        048B  5044        VBUSDATA        <^X1E>,1                ;89
   01   048C  5045        .BYTE   ^X01
        048D  5046        VBUSDATA        <^X1E>,1                ;8A
   01   048E  5047        .BYTE   ^X01
        048F  5048        VBUSDATA        <^X1E>,1                ;8B
   01   0490  5049        .BYTE   ^X01
        0491  5050        VBUSDATA        <^X1E>,1                ;8C
   01   0492  5051        .BYTE   ^X01
        0493  5052        VBUSDATA        <^X1E>,1                ;8D
   01   0494  5053        .BYTE   ^X01
        0495  5054        VBUSDATA        <^X1E>,1                ;8E
   01   0496  5055        .BYTE   ^X01
        0497  5056        VBUSDATA        <^X1E>,1                ;8F
   01   0498  5057        .BYTE   ^X01
        0499  5058        VBUSDATA        <^X1E>,1                ;90
   01   049A  5059        .BYTE   ^X01
        049B  5060        VBUSDATA        <^X1E>,1                ;91
   01   049C  5061        .BYTE   ^X01
        049D  5062        VBUSDATA        <^X1E>,1                ;92
   01   049E  5063        .BYTE   ^X01
        049F  5064        VBUSDATA        <^X1E>,1                ;93
   01   04A0  5065        .BYTE   ^X01
        04A1  5066        VBUSDATA        <^X1E>,1                ;94
   01   04A2  5067        .BYTE   ^X01
        04A3  5068        VBUSDATA        <^X1E>,1                ;95
   01   04A4  5069        .BYTE   ^X01
        04A5  5070        VBUSDATA        <^X1E>,1                ;96
   01   04A6  5071        .BYTE   ^X01
        04A7  5072        VBUSDATA        <^X1E>,1                ;97
   01   04A8  5073        .BYTE   ^X01
        04A9  5074        VBUSDATA        <^X1E>,1                ;98
   01   04AA  5075        .BYTE   ^X01
        04AB  5076        VBUSDATA        <^X1E>,1                ;99
   01   04AC  5077        .BYTE   ^X01
        04AD  5078        VBUSDATA        <^X1E>,1                ;9A
   01   04AE  5079        .BYTE   ^X01
        04AF  5080        VBUSDATA        <^X1E>,1                ;9B
   01   04B0  5081        .BYTE   ^X01
        04B1  5082        VBUSDATA        <^X1E>,1                ;9C
```

```
                    01  04B2  5083          .BYTE   ^X01
                        04B3  5084          VBUSDATA        <^X1E>,1                ;9D
                    01  04B4  5085          .BYTE   ^X01
                        04B5  5086          VBUSDATA        <^X1E>,1                ;9E
                    01  04B6  5087          .BYTE   ^X01
                        04B7  5088          VBUSDATA        <^X1E>,1                ;9F
                    01  04B8  5089          .BYTE   ^X01
                        04B9  5090          VBUSDATA        <^X1E>,1                ;A0
                    01  04BA  5091          .BYTE   ^X01
                        04BB  5092          VBUSDATA        <^X1E>,1                ;A1
                    01  04BC  5093          .BYTE   ^X01
                        04BD  5094          VBUSDATA        <^X1E>,1                ;A2
                    01  04BE  5095          .BYTE   ^X01
                        04BF  5096          VBUSDATA        <^X1E>,1                ;A3
                    01  04C0  5097          .BYTE   ^X01
                        04C1  5098          VBUSDATA        <^X1E>,1                ;A4
                    01  04C2  5099          .BYTE   ^X01
                        04C3  5100          VBUSDATA        <^X1E>,1                ;A5
                    01  04C4  5101          .BYTE   ^X01
                        04C5  5102          VBUSDATA        <^X1E>,1                ;A6
                    01  04C6  5103          .BYTE   ^X01
                        04C7  5104          VBUSDATA        <^X1E>,1                ;A7
                    01  04C8  5105          .BYTE   ^X01
                        04C9  5106          VBUSDATA        <^X1E>,1                ;A8
                    01  04CA  5107          .BYTE   ^X01
                        04CB  5108          VBUSDATA        <^X1E>,1                ;A9
                    01  04CC  5109          .BYTE   ^X01
                        04CD  5110          VBUSDATA        <^X1E>,1                ;AA
                    01  04CE  5111          .BYTE   ^X01
                        04CF  5112          VBUSDATA        <^X1E>,1                ;AB
                    01  04D0  5113          .BYTE   ^X01
                        04D1  5114          VBUSDATA        <^X1E>,1                ;AC
                    01  04D2  5115          .BYTE   ^X01
                        04D3  5116          VBUSDATA        <^X1E>,1                ;AD
                    01  04D4  5117          .BYTE   ^X01
                        04D5  5118          VBUSDATA        <^X1E>,1                ;AE
                    01  04D6  5119          .BYTE   ^X01
                        04D7  5120          VBUSDATA        <^X1E>,1                ;AF
                    01  04D8  5121          .BYTE   ^X01
                        04D9  5122          VBUSDATA        <^X1E>,1                ;B0
                    01  04DA  5123          .BYTE   ^X01
                        04DB  5124          VBUSDATA        <^X1E>,1                ;B1
                    01  04DC  5125          .BYTE   ^X01
                        04DD  5126          VBUSDATA        <^X1E>,1                ;B2
                    01  04DE  5127          .BYTE   ^X01
                        04DF  5128          VBUSDATA        <^X1E>,1                ;B3
                    01  04E0  5129          .BYTE   ^X01
                        04E1  5130          VBUSDATA        <^X1E>,1                ;B4
                    01  04E2  5131          .BYTE   ^X01
                        04E3  5132          VBUSDATA        <^X1E>,1                ;B5
                    01  04E4  5133          .BYTE   ^X01
                        04E5  5134          VBUSDATA        <^X1E>,1                ;B6
                    01  04E6  5135          .BYTE   ^X01
                        04E7  5136          VBUSDATA        <^X1E>,1                ;B7
                    01  04E8  5137          .BYTE   ^X01
```

```
                    04E9  5138          VBUSDATA        <^X1E>,1                ;B8
01                  04EA  5139          .BYTE   ^X01
                    04EB  5140          VBUSDATA        <^X1E>,1                ;B9
01                  04EC  5141          .BYTE   ^X01
                    04ED  5142          VBUSDATA        <^X1E>,1                ;BA
01                  04EE  5143          .BYTE   ^X01
                    04EF  5144          VBUSDATA        <^X1E>,1                ;BB
01                  04F0  5145          .BYTE   ^X01
                    04F1  5146          VBUSDATA        <^X1E>,1                ;BC
01                  04F2  5147          .BYTE   ^X01
                    04F3  5148          VBUSDATA        <^X1E>,1                ;BD
01                  04F4  5149          .BYTE   ^X01
                    04F5  5150          VBUSDATA        <^X1E>,1                ;BE
01                  04F6  5151          .BYTE   ^X01
                    04F7  5152          VBUSDATA        <^X1E>,1                ;BF
01                  04F8  5153          .BYTE   ^X01
                    04F9  5154          VBUSDATA        <^X1E>,1                ;C0
01                  04FA  5155          .BYTE   ^X01
                    04FB  5156          VBUSDATA        <^X1E>,1                ;C1
01                  04FC  5157          .BYTE   ^X01
                    04FD  5158          VBUSDATA        <^X1E>,1                ;C2
01                  04FE  5159          .BYTE   ^X01
                    04FF  5160          VBUSDATA        <^X1E>,1                ;C3
01                  0500  5161          .BYTE   ^X01
                    0501  5162          VBUSDATA        <^X1E>,1                ;C4
01                  0502  5163          .BYTE   ^X01
                    0503  5164          VBUSDATA        <^X1E>,1                ;C5
01                  0504  5165          .BYTE   ^X01
                    0505  5166          VBUSDATA        <^X1E>,1                ;C6
01                  0506  5167          .BYTE   ^X01
                    0507  5168          VBUSDATA        <^X1E>,1                ;C7
01                  0508  5169          .BYTE   ^X01
                    0509  5170          VBUSDATA        <^X1E>,1                ;C8
01                  050A  5171          .BYTE   ^X01
                    050B  5172          VBUSDATA        <^X1E>,1                ;C9
01                  050C  5173          .BYTE   ^X01
                    050D  5174          VBUSDATA        <^X1E>,1                ;CA
01                  050E  5175          .BYTE   ^X01
                    050F  5176          VBUSDATA        <^X1E>,1                ;CB
01                  0510  5177          .BYTE   ^X01
                    0511  5178          VBUSDATA        <^X1E>,1                ;CC
01                  0512  5179          .BYTE   ^X01
                    0513  5180          VBUSDATA        <^X1E>,1                ;CD
01                  0514  5181          .BYTE   ^X01
                    0515  5182          VBUSDATA        <^X1E>,1                ;CE
01                  0516  5183          .BYTE   ^X01
                    0517  5184          VBUSDATA        <^X1E>,1                ;CF
01                  0518  5185          .BYTE   ^X01
                    0519  5186          VBUSDATA        <^X1E>,1                ;D0
01                  051A  5187          .BYTE   ^X01
                    051B  5188          VBUSDATA        <^X1E>,1                ;D1
01                  051C  5189          .BYTE   ^X01
                    051D  5190          VBUSDATA        <^X1E>,1                ;D2
01                  051E  5191          .BYTE   ^X01
                    051F  5192          VBUSDATA        <^X1E>,1                ;D3
```

```
01  0520  5193          .BYTE   ^X01
    0521  5194          VBUSDATA        <^X1E>,1                ;D4
01  0522  5195          .BYTE   ^X01
    0523  5196          VBUSDATA        <^X1E>,1                ;D5
01  0524  5197          .BYTE   ^X01
    0525  5198          VBUSDATA        <^X1E>,1                ;D6
01  0526  5199          .BYTE   ^X01
    0527  5200          VBUSDATA        <^X1E>,1                ;D7
01  0528  5201          .BYTE   ^X01
    0529  5202          VBUSDATA        <^X1E>,1                ;D8
01  052A  5203          .BYTE   ^X01
    052B  5204          VBUSDATA        <^X1E>,1                ;D9
01  052C  5205          .BYTE   ^X01
    052D  5206          VBUSDATA        <^X1E>,1                ;DA
01  052E  5207          .BYTE   ^X01
    052F  5208          VBUSDATA        <^X1E>,1                ;DB
01  0530  5209          .BYTE   ^X01
    0531  5210          VBUSDATA        <^X1E>,1                ;DC
01  0532  5211          .BYTE   ^X01
    0533  5212          VBUSDATA        <^X1E>,1                ;DD
01  0534  5213          .BYTE   ^X01
    0535  5214          VBUSDATA        <^X1E>,1                ;DE
01  0536  5215          .BYTE   ^X01
    0537  5216          VBUSDATA        <^X1E>,1                ;DF
01  0538  5217          .BYTE   ^X01
    0539  5218          VBUSDATA        <^X1E>,1                ;E0
01  053A  5219          .BYTE   ^X01
    053B  5220          VBUSDATA        <^X1E>,1                ;E1
01  053C  5221          .BYTE   ^X01
    053D  5222          VBUSDATA        <^X1E>,1                ;E2
01  053E  5223          .BYTE   ^X01
    053F  5224          VBUSDATA        <^X1E>,1                ;E3
01  0540  5225          .BYTE   ^X01
    0541  5226          VBUSDATA        <^X1E>,1                ;E4
01  0542  5227          .BYTE   ^X01
    0543  5228          VBUSDATA        <^X1E>,1                ;E5
01  0544  5229          .BYTE   ^X01
    0545  5230          VBUSDATA        <^X1E>,1                ;E6
01  0546  5231          .BYTE   ^X01
    0547  5232          VBUSDATA        <^X1E>,1                ;E7
01  0548  5233          .BYTE   ^X01
    0549  5234          VBUSDATA        <^X1E>,1                ;E8
01  054A  5235          .BYTE   ^X01
    054B  5236          VBUSDATA        <^X1E>,1                ;E9
01  054C  5237          .BYTE   ^X01
    054D  5238          VBUSDATA        <^X1E>,1                ;EA
01  054E  5239          .BYTE   ^X01
    054F  5240          VBUSDATA        <^X1E>,1                ;EB
01  0550  5241          .BYTE   ^X01
    0551  5242          VBUSDATA        <^X1E>,1                ;EC
01  0552  5243          .BYTE   ^X01
    0553  5244          VBUSDATA        <^X1E>,1                ;ED
01  0554  5245          .BYTE   ^X01
    0555  5246          VBUSDATA        <^X1E>,1                ;EE
01  0556  5247          .BYTE   ^X01
```

```
                        0557  5248          VBUSDATA        <^X1E>,1                ;EF
                  01    0558  5249          .BYTE   ^X01
                        0559  5250          VBUSDATA        <^X1E>,1                ;F0
                  01    055A  5251          .BYTE   ^X01
                        055B  5252          VBUSDATA        <^X1E>,1                ;F1
                  01    055C  5253          .BYTE   ^X01
                        055D  5254          VBUSDATA        <^X1E>,1                ;F2
                  01    055E  5255          .BYTE   ^X01
                        055F  5256          VBUSDATA        <^X1E>,1                ;F3
                  01    0560  5257          .BYTE   ^X01
                        0561  5258          VBUSDATA        <^X1E>,1                ;F4
                  01    0562  5259          .BYTE   ^X01
                        0563  5260          VBUSDATA        <^X1E>,1                ;F5
                  01    0564  5261          .BYTE   ^X01
                        0565  5262          VBUSDATA        <^X1E>,1                ;F6
                  01    0566  5263          .BYTE   ^X01
                        0567  5264          VBUSDATA        <^X1E>,1                ;F7
                  01    0568  5265          .BYTE   ^X01
                        0569  5266          VBUSDATA        <^X1E>,1                ;F8
                  01    056A  5267          .BYTE   ^X01
                        056B  5268          VBUSDATA        <^X1E>,1                ;F9
                  01    056C  5269          .BYTE   ^X01
                        056D  5270          VBUSDATA        <^X1E>,1                ;FA
                  01    056E  5271          .BYTE   ^X01
                        056F  5272          VBUSDATA        <^X1E>,1                ;FB
                  01    0570  5273          .BYTE   ^X01
                        0571  5274          VBUSDATA        <^X1E>,1                ;FC
                  01    0572  5275          .BYTE   ^X01
                        0573  5276          VBUSDATA        <^X1E>,1                ;FD
                  01    0574  5277          .BYTE   ^X01
                        0575  5278          VBUSDATA        <^X1E>,1                ;FE
                  01    0576  5279          .BYTE   ^X01
                        0577  5280          VBUSDATA        <^X1E>,1                ;FF
                        0578  5281
                        0578  5282 BEGIN_CPU_DATA
                        0002  5283
                        0002  5284 DCS_DATA
                        0001  5285
U 00. 7700.C800.0364.0300.0470.1801     0001  5286 ;% 00    NOP                                     ; GET UP TO SPEED
U 01. 5700.C840.0364.0304.0470.1802     000C  5290 ;% 01    R[TEMP1]_RDM                            ; LOAD THE OPCODE INTO RTEMP1
U 02. 7700.F800.57D7.D7FF.8470.1803     0017  5294 ;% 02    LONLIT_[50505000]                       ; LOAD THE OPERANDS INTO THE LONLIT REG
U 03. 7700.8800.5BE5.03D4.0470.1804     0022  5298 ;% 03    Q_R[LONLIT]                             ; MOVE THE OPERANDS TO THE Q REGISTER
U 04. 7700.0841.00A4.0304.0470.1805     002D  5302 ;% 04    R[TEMP1]_M[TEMP1].OR.Q                  ; OR THE OPCODE WITH THE OPERANDS
U 05. 7700.4800.5BE4.03D8.4A70.1806     0038  5306 ;% 05    VA_R[ZERO]                              ; ZERO THE VA
U 06. 7701.C800.5BE4.0204.5590.1807     0043  5310 ;% 06    WRITE.LONG R[TEMP1]                     ; WRITE THE MODEL INSTRUCTION TO CACHE
U 07. 7700.8800.5BE4.0304.5470.1808     004E  5314 ;% 07    WDR_UNROT(R[TEMP1])                     ; CMI OFF SO HOLD DATA FOR SECOND CYCLE
U 08. 7700.C800.0364.0300.4470.1809     0059  5318 ;% 08    VA_VA+4                                 ; INCREMENT THE VA
U 09. 7701.0800.5BE4.02D8.5590.180A     0064  5322 ;% 09    WRITE.LONG R[ZERO]                      ; VALIDATE NEXT LOCATION FOR XB1
U 0A. 7700.C800.5BE4.03D8.5470.180B     006F  5326 ;% 0A    WDR_UNROT(R[ZERO])                      ; CMI OFF SO HOLD DATA FOR SECOND CYCLE
U 0B. 7700.C800.5BE4.03D8.4870.180C     007A  5330 ;% 0B    PC_R[ZERO]                              ; START INSTRUCTION PREFETCH, LOAD XB'S
U 0C. 7701.0857.5924.0202.0470.180D     0085  5334 ;% 0C    R[TEMP0]_M[XB.PC_PC+I],ISIZE[LONG]      ; LOAD RTEMP0 WITH XB0
U 0D. 7700.4800.5BE4.03D8.4870.180E     0090  5338 ;% 0D    PC_R[ZERO]                              ; RESET THE PC AND RELOAD THE XB'S
U 0E. 7600.0800.0364.1300.0470.180F     009B  5342 ;% 0E    BUT/IRD1,STROBE.V.BUS                   ; EXECUTE IRD1 AND STROBE VBUS
U 0F. 7300.C800.0364.0300.0470.1810     00A6  5346 ;% 0F    STOP.CLOCK                              ; HALT
                                        00B1  5350
```

```
U 10. 7700,4800,0364,0300,0470,1811   00B1   5351 ;% 10   NOP                                  ; GET CPU UP TO SPEED
U 11. 5700,4800,0364,0300,2A70,1812   00BC   5355 ;% 11   WCTRL/FPA.ENABLE_WB5,WB_RDM          ; WRITE THE FPA'S ENABLE BIT
U 12. 7700,C800,0364,0300,0470,1813   00C7   5359 ;% 12   NOP                                  ; NOP TO GET CPU UP TO SPEED
U 13. 7700,4800,0364,C300,0470,1800   00D2   5363 ;% 13   BUT/NO.FPA,NEXT/1800                 ; CHECK FPA STATUS
U 14. 7300,C800,0364,0300,0470,1815   00DD   5367 ;% 14   STOP.CLOCK                           ; HALT
                                      00E8   5371
U 15. 7700,C800,0364,0300,0470,1816   00E8   5372 ;% 15   NOP                                  ; NOP TO GET CPU UP TO SPEED
U 16. 7700,B800,7BFF,FFFF,8470,1817   00F3   5376 ;% 16   LONLIT_[08000000]                    ; LOAD LONGLIT WITH DATA TO SET FPD
U 17. 7700,8800,5BF4,03D4,0070,1818   00FE   5380 ;% 17   PSL_R[LONLIT]                        ; SET FPD BIT IN PSL
U 18. 7700,7800,7F7F,7F7F,0470,1819   0109   5384 ;% 18   LONLIT_[01010101]                    ; LOAD LONGLIT WITH 4 NOP INSTRUCTIONS
U 19. 7700,C800,5BE4,03D8,4A70,181A   0114   5388 ;% 19   VA_R[ZERO]                           ; ZERO THE VA
U 1A. 7701,0800,5BE4,02D4,5590,181B   011F   5392 ;% 1A   WRITE.LONG R[LONLIT]                 ; WRITE NOPS TO CACHE FOR XB0
U 1B. 7700,C800,5BE4,03D4,5470,181C   012A   5396 ;% 1B   WDR_UNROT(R[LONLIT])                 ; CMI OFF SO HOLD FOR SECOND CYCLE
U 1C. 7700,C800,0364,0300,4470,181D   0135   5400 ;% 1C   VA_VA+4                             ; INCREMENT THE VA
U 1D. 7701,0800,5BE4,02D4,5590,181E   0140   5404 ;% 1D   WRITE.LONG R[LONLIT]                 ; VALIDATE NEXT LOCATION FOR XB1
U 1E. 7700,C800,5BE4,03D4,5470,181F   014B   5408 ;% 1E   WDR_UNROT(R[LONLIT])                 ; CMI OFF SO HOLD FOR SECOND CYCLE
U 1F. 7700,4800,5BE4,03D8,4870,1820   0156   5412 ;% 1F   PC_R[ZERO]                           ; ZERO THE PC TO LOAD XB'S
U 20. 7700,4800,0364,1700,0470,1821   0161   5416 ;% 20   IRD1TEST                            ; IRD1 TO SET FPA FOR NON-FPA OPCODE
U 21. 7300,4800,0364,0300,0470,1822   016C   5420 ;% 21   STOP.CLOCK                           ; HALT
                                      0177   5424
                                      0177   5425 RTEMP_DATA
                                      0001   5426
                           00000000   0001   5427         .LONG   0       ; RTEMP0 HOLDS THE CONTENTS OF XB0
                           00000000   0005   5428         .LONG   0       ; RTEMP1 HOLDS THE INSTRUCTION OPCODE
                                      0009   5429
                                      0009   5430 END_CPU_DATA
                                      0578   5431
                                      0578   5432 END_TEST_DATA
                                      0578
                                      0578         ;-------------------------------------------------------------------------
                                      0578   5433
                                      0578   5434 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:   TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
                        0031        .SBTTL  "TEST   0C6     First IRDX Instruction Address Decode Test 1
                        0031  5436 ;*******************************************************************************
                        0031  5437 ;++
                        0031  5438 ;
                        0031  5439 ; FUNCTIONAL DESCRIPTION
                        0031  5440 ;
                        0031  5441 ;       This test will examine the Native Mode IRDX roms when the IRD Counter
                        0031  5442 ;       is equal to zero and the operand specifier is for a memory reference.
                        0031  5443 ;
                        0031  5444 ; TEST ALGORITHM
                        0031  5445 ;
                        0031  5446 ;       This test consists of three subtests.  The first subtest will examine
                        0031  5447 ;       the IRDX roms when the Floating Point Accelerator (FPA) is not present
                        0031  5448 ;       in the system.  The second subtest will examine the IRDX roms when the
                        0031  5449 ;       FPA is present but disabled.  The third subtest will examine the IRDX
                        0031  5450 ;       roms when the FPA is present and enabled.  The algorithm is as follows:
                        0031  5451 ;
                        0031  5452 ;        1.  Initialize the processor.
                        0031  5453 ;        2.  Disable CTRL/P interrupts.
                        0031  5454 ;        3.  Check to see if FPA present
                        0031  5455 ;               If FPA present
                        0031  5456 ;                   Then skip to Subtest 2
                        0031  5457 ;               Else fall through to Subtest 1
                        0031  5458 ;
                        0031  5459 ;       Subtest 1
                        0031  5460 ;
                        0031  5461 ;        4.  Set-up flag to expect a micro trap.
                        0031  5462 ;        5.  Initialize loop I for 256 opcodes (00-FF).
                        0031  5463 ;        6.  Store the index I in a location that will be used to load into the
                        0031  5464 ;            WDREG.  The value is used as the opcode.
                        0031  5465 ;        7.  Loop on error from here.
                        0031  5466 ;        8.  Execute the following test microcode:
                        0031  5467 ;               a.  NOP to get the RDM up to speed.
                        0031  5468 ;               b.  Read the WDREG into RTEMP2 (opcode).
                        0031  5469 ;               c.  Load the operands into the LONLIT register.
                        0031  5470 ;               d.  Store the operands into the Q register to prepare for ORing
                        0031  5471 ;                   the opcode with the operands.
                        0031  5472 ;               e.  OR the opcode and operands together (XB0 data).
                        0031  5473 ;               f.  Store XB0 data in M[TEMP8] in case needed for error callout.
                        0031  5474 ;               g.  Zero the VA.
                        0031  5475 ;               h.  Validate cache location 0 by first writing the data in
                        0031  5476 ;                   RTEMP2 then reading the data back.
                        0031  5477 ;               i.  Update the VA by 4.
                        0031  5478 ;               j.  Load the second operand data into the LONLIT register
                        0031  5479 ;                   (XB1 data).
                        0031  5480 ;               k.  Store XB1 data in M[TEMP9] in case needed for error callout.
                        0031  5481 ;               l.  Validate cache location 4 by first writing the data in
                        0031  5482 ;                   the LONLIT register then reading the data back.
                        0031  5483 ;               m.  Load the PC with zero to cause a prefetch to take place.
                        0031  5484 ;               n.  Do an IRD1TST to init the IRD counter and prepare for
                        0031  5485 ;                   instruction/operand specifier evaluation.
                        0031  5486 ;               o.  Halt.
                        0031  5487 ;        9.  Perform the following test microcode.
                        0031  5488 ;               a.  NOP to get the RDM up to speed.
                        0031  5489 ;               b.  Do an IRDX.
```

```
0031  5490 ;                    c.  Halt.
0031  5491 ;          10.  Compare the contents of the CSTRP register (output from the IRDX
0031  5492 ;               Roms) against the Rom Table.
0031  5493 ;          11.  If the two do not equal, execute the following microcode and
0031  5494 ;               signal an error (this is done to set-up for printing RTEMP0 & 1
0031  5495 ;               on the error callout):
0031  5496 ;                  a.  NOP to get the RDM up to speed.
0031  5497 ;                  b.  Load RTEMP0 with the contents of M[TEMP8]
0031  5498 ;                  c.  Load RTEMP1 with the contents of M[TEMP9]
0031  5499 ;                  d.  Halt
0031  5500 ;          12.  Continue with loop I for the 256 VAX opcodes.
0031  5501 ;          13.  Re-enable CTRL/P interrupts then skip to next tests.
0031  5502 ;
0031  5503 ;          Subtest 2
0031  5504 ;
0031  5505 ;          14.  Load the FPA disable bit (bit <5> equal to 0).
0031  5506 ;          15.  Loop on error from here.
0031  5507 ;          16.  Set-up to expect micro trap.
0031  5508 ;          17.  Execute the test microcode.  This is done only to get the FPA to
0031  5509 ;               a state where it can be disabled or enabled.  If this had not been
0031  5510 ;               done before then the FPA would be setting in the initialize state
0031  5511 ;               which forces FPA enable.
0031  5512 ;          18.  Execute the following microcode to disable the FPA:
0031  5513 ;                  a.  NOP to get the RDM up to speed.
0031  5514 ;                  b.  Read the RDM WDREG onto the W Bus and then write the W Bus
0031  5515 ;                      to the FPA.
0031  5516 ;                  c.  Perform a NO.FPA test to insure that the FPA got disabled.
0031  5517 ;                  d.  Halt.
0031  5518 ;          19.  Check the Trap register to insure that the FPA got disabled.
0031  5519 ;          20.  If the FPA is not disabled, then signal an error.
0031  5520 ;          21.  Set-up flag to expect a micro trap.
0031  5521 ;          22.  Initialize loop I for 256 opcodes (00-FF).
0031  5522 ;          23.  Store the index I in a location that will be used to load into the
0031  5523 ;               WDREG.  The value is used as the opcode.
0031  5524 ;          24.  Loop on error from here.
0031  5525 ;          25.  Execute the following test microcode:
0031  5526 ;                  a.  NOP to get the RDM up to speed.
0031  5527 ;                  b.  Read the WDREG into RTEMP2 (opcode).
0031  5528 ;                  c.  Load the operands into the LONLIT register.
0031  5529 ;                  d.  Store the operands into the Q register to prepare for ORing
0031  5530 ;                      the opcode with the operands.
0031  5531 ;                  e.  OR the opcode and operands together (XB0 data).
0031  5532 ;                  f.  Store XB0 data in M[TEMP8] in case needed for error callout.
0031  5533 ;                  g.  Zero the VA.
0031  5534 ;                  h.  Validate cache location 0 by first writing the data in
0031  5535 ;                      RTEMP2 then reading the data back.
0031  5536 ;                  i.  Update the VA by 4.
0031  5537 ;                  j.  Load the second operand data into the LONLIT register
0031  5538 ;                      (XB1 data).
0031  5539 ;                  k.  Store XB1 data in M[TEMP9] in case needed for error callout.
0031  5540 ;                  l.  Validate cache location 4 by first writing the data in
0031  5541 ;                      the LONLIT register then reading the data back.
0031  5542 ;                  m.  Load the PC with zero to cause a prefetch to take place.
0031  5543 ;                  n.  Do an IRD1 to init the IRD counter and prepare for instruc-
0031  5544 ;                      tion/operand specifier evaluation.
```

```
0031  5545 ;                   o.  Halt.
0031  5546 ;          26.  Perform the following test microcode.
0031  5547 ;                   a.  NOP to get the RDM up to speed.
0031  5548 ;                   b.  Do an IRDX.
0031  5549 ;                   c.  Halt.
0031  5550 ;          27.  Compare the contents of the CSTRP register (output from the IRDX
0031  5551 ;               Roms) against the Rom Table.
0031  5552 ;          28.  If the two do not equal, execute the following microcode and
0031  5553 ;               signal an error (this is done to set-up for printing RTEMP0 & 1
0031  5554 ;               on the error callout):
0031  5555 ;                   a.  NOP to get the RDM up to speed.
0031  5556 ;                   b.  Load RTEMP0 with the contents of M[TEMP8]
0031  5557 ;                   c.  Load RTEMP1 with the contents of M[TEMP9]
0031  5558 ;                   d.  Halt
0031  5559 ;          29.  Continue with loop I for the 256 VAX opcodes.
0031  5560 ;          30.  Go to Subtest 3.
0031  5561 ;
0031  5562 ;          Subtest 3
0031  5563 ;
0031  5564 ;          31.  Load the FPA enable bit (bit <5> equal to 1).
0031  5565 ;          32.  Loop on error from here.
0031  5566 ;          33.  Set-up to expect micro trap.
0031  5567 ;          34.  Execute the following microcode to enable the FPA:
0031  5568 ;                   a.  NOP to get the RDM up to speed.
0031  5569 ;                   b.  Read the RDM WDREG onto the W Bus and then write the W Bus
0031  5570 ;                       to the FPA.
0031  5571 ;                   c.  Perform a NO.FPA test to insure that the FPA got enabled.
0031  5572 ;                   d.  Halt.
0031  5573 ;          35.  Check the Trap register to insure that the FPA got enabled.
0031  5574 ;          36.  If the FPA is not enabled, then signal an error.
0031  5575 ;          37.  Set-up flag to expect a micro trap.
0031  5576 ;          38.  Initialize loop I for 256 opcodes (00-FF).
0031  5577 ;          39.  Store the index I in a location that will be used to load into the
0031  5578 ;               WDREG.  The value is used as the opcode.
0031  5579 ;          40.  Loop on error from here.
0031  5580 ;          41.  Set then Clear the DCLO bit in the RDCTRL register to perform an
0031  5581 ;               initialize.  This is done because when the test microcode
0031  5582 ;               executes, if the opcode was an FP opcode then there is a
0031  5583 ;               possibility that the processor could be setting there in an
0031  5584 ;               FPAWAIT state (M clock and D clock stalled).  If this were to
0031  5585 ;               happen then the cpu clocks would no longer run and the remaining
0031  5586 ;               test opcodes would fail.
0031  5587 ;          42.  Execute the following test microcode:
0031  5588 ;                   a.  NOP to get the RDM up to speed.
0031  5589 ;                   b.  Read the WDREG into RTEMP2 (opcode).
0031  5590 ;                   c.  Load the operands into the LONLIT register.
0031  5591 ;                   d.  Store the operands into the Q register to prepare for ORing
0031  5592 ;                       the opcode with the operands.
0031  5593 ;                   e.  OR the opcode and operands together (XB0 data).
0031  5594 ;                   f.  Store XB0 data in M[TEMP8] in case needed for error callout.
0031  5595 ;                   g.  Zero the VA.
0031  5596 ;                   h.  Validate cache location 0 by first writing the data in
0031  5597 ;                       RTEMP2 then reading the data back.
0031  5598 ;                   i.  Update the VA by 4.
0031  5599 ;                   j.  Load the second operand data into the LONLIT register
```

```
0031 ;                    (XB1 data).
0031 ;                k.  Store XB1 data in M[TEMP9] in case needed for error callout.
0031 ;                l.  Validate cache location 4 by first writing the data in
0031 ;                    the LONLIT register then reading the data back.
0031 ;                m.  Load the PC with zero to cause a prefetch to take place.
0031 ;                n.  Do an IRD1 to init the IRD counter and prepare for instruc-
0031 ;                    tion/operand specifier evaluation.
0031 ;                o.  Halt.
0031 ;        43.  Perform the following test microcode.
0031 ;                a.  NOP to get the RDM up to speed.
0031 ;                b.  Do an IRDX.
0031 ;                c.  Halt.
0031 ;        44.  Compare the contents of the CSTRP register (output from the IRDX
0031 ;             Roms) against the Rom Table.
0031 ;        45.  If the two do not equal, execute the following microcode and
0031 ;             signal an error (this is done to set-up for printing RTEMP0 & 1
0031 ;             on the error callout):
0031 ;                a.  NOP to get the RDM up to speed.
0031 ;                b.  Load RTEMP0 with the contents of M[TEMP8]
0031 ;                c.  Load RTEMP1 with the contents of M[TEMP9]
0031 ;                d.  Halt
0031 ;        46.  Continue with loop I for the 256 VAX opcodes.
0031 ;        47.  Re-enable CTRL/P interrupts.
0031 ;        48.  Skip to next tests.
0031 ;
0031 ; TEST PATTERNS
0031 ;
0031 ;        For this test the opcodes are generated by a simple count pattern.  The
0031 ;        opcodes are based on the 256 base machine instructions as specified in
0031 ;        the VAX architecture handbook.  The generated opcodes are written to
0031 ;        the CPU and combined (ORed) with three bytes of operand data to form
0031 ;        a machine instruction.  The operands all specify Register Deferred mode
0031 ;        using registers 0-6 and are loaded from the LONLIT register.
0031 ;
0031 ;        Also required are two tables of expected control store addresses for
0031 ;        the first IRDX cycle.  One table contains the expected addresses for
0031 ;        the first IRDX when the FPA is disabled, the other for when it is
0031 ;        enabled.
0031 ;
0031 ; LOGIC DESCRIPTION
0031 ;
0031 ;        This test will check the validity of the IRDX roms on the DPM module.
0031 ;        The IRDX roms are addressed by bits <7:0> of the Instruction Register.
0031 ;        Along with being addressed by the IR, there are three other signals
0031 ;        that affect the address.  They are IRD CTR 0 H, FPA ENABLED L, and
0031 ;        REG MODE H.  This test will examine every 8th(hex) address starting at
0031 ;        address 2 with FPA disabled and starting at address 0 with FPA enabled.
0031 ;
0031 ; ASSUMPTIONS
0031 ;
0031 ;        This test assumes the following logic has already been tested; the
0031 ;        CPU's data path, the cache, the XB, and the IR and OSR.
0031 ;
0031 ; ERROR DESCRIPTION
0031 ;
```

```
0031  5655 ;       Subtest 1
0031  5656 ;
0031  5657 ;               Expected CS Address
0031  5658 ;               Received CS Address
0031  5659 ;
0031  5660 ;       Subtest 2
0031  5661 ;
0031  5662 ;               First IFERROR
0031  5663 ;
0031  5664 ;                       Expected CS Address
0031  5665 ;                       Received CS Address
0031  5666 ;
0031  5667 ;               Second IFERROR
0031  5668 ;
0031  5669 ;                       Expected IRDX Rom Contents
0031  5670 ;                       Received IRDX Rom Contents
0031  5671 ;                       Loop I count
0031  5672 ;                       Contents of XB0
0031  5673 ;                       Contents of XB1
0031  5674 ;
0031  5675 ;       Subtest 3
0031  5676 ;
0031  5677 ;               First IFERROR
0031  5678 ;
0031  5679 ;                       Expected CS Address
0031  5680 ;                       Received CS Address
0031  5681 ;
0031  5682 ;               Second IFERROR
0031  5683 ;
0031  5684 ;                       Expected IRDX Rom Contents
0031  5685 ;                       Received IRDX Rom Contents
0031  5686 ;                       Loop I count
0031  5687 ;                       Contents of XB0
0031  5688 ;                       Contents of XB1
0031  5689 ;--
0031  5690 ;*******************************************************************************
0031  5691
0031  5692         INITIALIZE                                  ; initialize the comet cpu
0033  5693         FETCH          <^X19>                       ; disable ctrl-p interrupts
0038  5694         BRSTCLK        <^X1E>                       ; ...
003C  5695         FETCH          <^X16>                       ; execute a but/no.fpa to check
0041  5696         BRSTCLK        <^X18>                       ; for fpa present
0045  5697         CMPREGMSK      CSTRP,1$,,2$,,W              ; is the fpa present?
0051  5698         SKIP           100$,ONERROR                 ; if so then skip to subtest 2
0058  5699
0058  5700         SUBTEST        1
005B
005B       ;////////////////////////////////////////////////////////////////////////////
005B  5701 ;+
005B  5702 ;
005B  5703 ; FPA is not present.
005B  5704 ;
005B  5705 ;-
005B  5706         EXPUTRAP                                    ; else expect non-dcs addresses
005D  5707         LOOP           I,1,256                      ; loop for 256 opcodes
```

```
                0064  5708          SAVEINDEX       I,9$                    ; save the loop count
                0069  5709          LOADREG         WDREG,9$,,L             ; load wdreg with loop count
                0071  5710          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                0073  5711          FETCH           0                       ; load the istrm data and do an
                0078  5712          BRSTCLK         <^X10>,INH              ; ird1tst
                007C  5713          FETCH           <^X11>                  ; execute the irdx
                0081  5714          BRSTCLK         <^X13>,INH              ; ...
                0085  5715          CMPREGMSK       CSTRP,30$,I,6$,,W       ; test the cs address
                0091  5716          SKIP            50$,NOERROR             ; Continue test if no error
                0098  5717          FETCH           <^X25>                  ; start dcs program
                009D  5718          BRSTCLK         <^X28>                  ; end dcs program
                00A1  5719          IFERROR         2,<<IRD>>,<<MIC>>       ; if bad address, signal error
                00A8  5720 50$:     ENDLOOP         I                       ; continue loop
                00AB  5721          FETCH           <^X1F>                  ; re-enable ctrl-p interrupts
                00B0  5722          BRSTCLK         <^X24>                  ; ...
                00B4  5723          SKIP                                    ; skip on to the next test
                00BB  5724
                00BB  5725 100$:    SUBTEST         2
                00BE
                00BE        ;////////////////////////////////////////////////////////////////////////////////
                00BE  5726 ;+
                00BE  5727 ;
                00BE  5728 ; FPA present but not enabled
                00BE  5729 ;
                00BE  5730 ;-
                00BE  5731          LOADREG         WDREG,7$,,L             ; load fpa disable bit
                00C6  5732          ERRLOOP                                 ; loop on error here
                00C8  5733          EXPUTRAP                                ; expect micro trap
                00CA  5734          FETCH           0                       ; start dcs program
                00CF  5735          BRSTCLK         <^X10>,INH              ; end dcs program
                00D3  5736          FETCH           <^X14>                  ; execute the disable
                00D8  5737          BRSTCLK         <^X18>,INH              ; and read back fpa status
                00DC  5738          CMPREGMSK       CSTRP,3$,,5$,,W         ; is the fpa disabled?
                00E8  5739          IFERROR         ,,<FPA>                 ; if not, signal an error
                00EE  5740          EXPUTRAP                                ; else expect non-dcs addresses
                00F0  5741          LOOP            I,1,256                 ; loop for 256 opcodes
                00F7  5742          SAVEINDEX       I,9$                    ; save the loop count
                00FC  5743          LOADREG         WDREG,9$,,L             ; load wdreg with loop count
                0104  5744          ERRLOOP                                 ; Loop on error from here
                0106  5745          FETCH           0                       ; execute a but/ird1tst
                010B  5746          BRSTCLK         <^X10>,INH              ; ...
                010F  5747          FETCH           <^X11>                  ; execute an irdx
                0114  5748          BRSTCLK         <^X13>,INH              ; ...
                0118  5749          CMPREGMSK       CSTRP,30$,I,6$,,W       ; is the cs address correct?
                0124  5750          SKIP            51$,NOERROR             ; continue testing if no error
                012B  5751          FETCH           <^X25>                  ; start dcs program
                0130  5752          BRSTCLK         <^X28>                  ; end dcs program
                0134  5753          IFERROR         2,<<IRD>>,<<MIC>>       ; if error then signal an error
                013B  5754 51$:     ENDLOOP         I                       ; continue loop
                013E  5755
                013E  5756          SUBTEST         3
                0141
                0141        ;////////////////////////////////////////////////////////////////////////////////
                0141  5757 ;+
                0141  5758 ;
```

```
                        0141  5759 ; FPA present and enabled
                        0141  5760 ;
                        0141  5761 ;-
                        0141  5762          LOADREG         WDREG,8$,,L             ; load fpa enable bit
                        0149  5763          ERRLOOP                                 ; loop on error from here
                        014B  5764          FETCH           <^X14>                  ; enable the fpa
                        0150  5765          BRSTCLK         <^X18>,INH              ; and read back fpa enable
                        0154  5766          CMPREGMSK       CSTRP,4$,,5$,,W         ; is the fpa enabled?
                        0160  5767          IFERROR         ,,<FPA>                 ; if not, signal an error
                        0166  5768          EXPUTRAP                                ; else expect non-dcs addresses
                        0168  5769          LOOP            I,1,256                 ; loop for 256 opcodes
                        016F  5770          SAVEINDEX       I,9$                    ; save the loop count
                        0174  5771          LOADREG         WDREG,9$,,L             ; load wdreg with loop count
                        017C  5772          ERRLOOP                                 ; Loop on error from here
                        017E  5773          LOADREG         RDCTRL,10$,,B           ; assert dclo
                        0186  5774          LOADREG         RDCTRL,11$,,B           ; de-assert dclo
                        018E  5775          FETCH           0                       ; execute a but/ird1tst
                        0193  5776          BRSTCLK         <^X10>,INH              ; ...
                        0197  5777          FETCH           <^X11>                  ; execute the but/irdx
                        019C  5778          BRSTCLK         <^X13>,INH              ; ...
                        01A0  5779          CMPREGMSK       CSTRP,31$,I,6$,,W       ; is the cs address correct?
                        01AC  5780          SKIP            52$,NOERROR             ; continue testing if no error
                        01B3  5781          FETCH           <^X25>                  ; start dcs program
                        01B8  5782          BRSTCLK         <^X28>                  ; end dcs program
                        01BC  5783          IFERROR         2,<<IRD>>,<<MIC>>       ; signal an error
                        01C3  5784 52$:     ENDLOOP         I                       ; continue loop
                        01C6  5785          FETCH           <^X1F>                  ; re-enable ctrl-p interrupts
                        01CB  5786          BRSTCLK         <^X24>                  ; ...
                        01CF  5787          SKIP                                    ; end of test
                        01D6  5788
                        01D6  5789 BEGIN_TEST_DATA
                        01D6
                        01D6       ;-----------------------------------------------------------------------
                        01D6  5790
                1802    01D6  5791 1$:      .WORD   ^X1802
                3FFE    01D8  5792 2$:      .WORD   ^X3FFE
                        01DA  5793
                1801    01DA  5794 3$:      .WORD   ^X1801
                1800    01DC  5795 4$:      .WORD   ^X1800
                3FFD    01DE  5796 5$:      .WORD   ^X3FFD
                        01E0  5797
                3FFF    01E0  5798 6$:      .WORD   ^X3FFF
                        01E2  5799
            00000000    01E2  5800 7$:      .LONG   0
            00000020    01E6  5801 8$:      .LONG   ^X00000020
                        01EA  5802
            00000000    01EA  5803 9$:      .LONG   0
                  40    01EE  5804 10$:     .BYTE   DC_LOW
                  00    01EF  5805 11$:     .BYTE   ^X00
                        01F0  5806
                        01F0  5807 ;
                        01F0  5808 ; Expected CS Addresses for Subtests 1 and 2
                        01F0  5809 ;
                        01F0  5810 ; The following is a table of all the expected control store addresses for the
                        01F0  5811 ; first IRDX.
```

```
                                        01F0  5812 ;
                                        01F0  5813
 03F8 03F8 03F8 03F8 03F8 03F8 03F8     01F0  5814 30$:     .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 00 - 06
 0108 0108 0381 0108 03BF 03BE 03F8     01FE  5815          .WORD   ^X03F8,^X03BE,^X03BF,^X0108,^X0381,^X0108,^X0108 ; 07 - 0D
 03F8 03F8 03F8 03F8 03F8 0158 0128     020C  5816          .WORD   ^X0128,^X0158,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 0E - 14
 03F8 03F8 03F8 03F8 0396 03A4 03F8     021A  5817          .WORD   ^X03F8,^X03A4,^X0396,^X03F8,^X03F8,^X03F8,^X03F8 ; 15 - 1B
 03B9 0288 03B9 03F8 03F8 03F8 03F8     0228  5818          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03B9,^X0288,^X03B9 ; 1C - 22
 03AE 03AE 0288 03BD 0288 03BC 0288     0236  5819          .WORD   ^X0288,^X03BC,^X0288,^X03BD,^X0288,^X03AE,^X03AE ; 23 - 29
 03F8 03B2 03B2 03AF 03AF 03B4 03B4     0244  5820          .WORD   ^X03B4,^X03B4,^X03AF,^X03AF,^X03B2,^X03B2,^X03F8 ; 2A - 30
 03B9 03BA 03B7 03B7 0158 00E5 03F8     0252  5821          .WORD   ^X03F8,^X00E5,^X0158,^X03B7,^X03B7,^X03BA,^X03B9 ; 31 - 37
 0158 0108 01DE 03B5 03B5 03B6 03C1     0260  5822          .WORD   ^X03C1,^X03B6,^X03B5,^X03B5,^X01DE,^X0108,^X0158 ; 38 - 3E
 0168 0118 0349 0345 0168 0118 0206     026E  5823          .WORD   ^X0206,^X0118,^X0168,^X0345,^X0349,^X0118,^X0168 ; 3F - 45
 00B2 0158 0158 0158 0158 0168 0118     027C  5824          .WORD   ^X0118,^X0168,^X0158,^X0158,^X0158,^X0158,^X00B2 ; 46 - 4C
 0329 0158 0168 0158 0168 0158 00B2     028A  5825          .WORD   ^X00B2,^X0158,^X0168,^X0158,^X0168,^X0158,^X0329 ; 4D - 53
 03F8 03F8 0158 03F8 001F 0108 0108     0298  5826          .WORD   ^X0108,^X0108,^X001F,^X03F8,^X0158,^X03F8,^X03F8 ; 54 - 5A
 00B4 00A8 0158 0158 0128 0128 03F8     02A6  5827          .WORD   ^X03F8,^X0128,^X0128,^X0158,^X0158,^X00A8,^X00B4 ; 5B - 61
 0068 00D8 00D4 00C8 00B8 0357 0356     02B4  5828          .WORD   ^X0356,^X0357,^X00B8,^X00C8,^X00D4,^X00D8,^X0068 ; 62 - 68
 0359 00B2 00B2 00B2 0074 0068 0068     02C2  5829          .WORD   ^X0068,^X0068,^X0074,^X00B2,^X00B2,^X00B2,^X0359 ; 69 - 6F
 0158 035A 00D2 032D 0158 0331 0158     02D0  5830          .WORD   ^X0158,^X0331,^X0158,^X032D,^X00D2,^X035A,^X0158 ; 70 - 76
 0158 022A 0178 0108 0178 0108 03F8     02DE  5831          .WORD   ^X03F8,^X0108,^X0178,^X0108,^X0178,^X022A,^X0158 ; 77 - 7D
 0118 0108 0118 0108 0118 020D 0158     02EC  5832          .WORD   ^X0158,^X020D,^X0118,^X0108,^X0118,^X0108,^X0118 ; 7E - 84
 0108 0118 0108 0118 02CD 02B9 0108     02FA  5833          .WORD   ^X0108,^X02B9,^X02CD,^X0118,^X0108,^X0118,^X0108 ; 85 - 8B
 0158 0108 0158 0108 0158 0108 0118     0308  5834          .WORD   ^X0118,^X0108,^X0158,^X0108,^X0158,^X0108,^X0158 ; 8C - 92
 00E5 00E5 0265 0243 008E 0221 0108     0316  5835          .WORD   ^X0108,^X0221,^X008E,^X0243,^X0265,^X00E5,^X00E5 ; 93 - 99
 0118 0206 0158 0108 0108 01DE 01DE     0324  5836          .WORD   ^X01DE,^X01DE,^X0108,^X0108,^X0158,^X0206,^X0118 ; 9A - A0
 0301 02BF 0108 0118 0108 0118 0108     0332  5837          .WORD   ^X0108,^X0118,^X0108,^X0118,^X0108,^X02BF,^X0301 ; A1 - A7
 0158 0108 0118 0108 0118 0108 0118     0340  5838          .WORD   ^X0118,^X0108,^X0118,^X0108,^X0118,^X0108,^X0158 ; A8 - AE
 008E 0221 0108 0158 0108 0158 0108     034E  5839          .WORD   ^X0108,^X0158,^X0108,^X0158,^X0108,^X0221,^X008E ; AF - B5
 03C6 03A6 03A7 029E 0292 0265 0243     035C  5840          .WORD   ^X0243,^X0265,^X0292,^X029E,^X03A7,^X03A6,^X03C6 ; B6 - BC
 0108 0118 0108 0118 03C9 03C8 03C7     036A  5841          .WORD   ^X03C7,^X03C8,^X03C9,^X0118,^X0108,^X0118,^X0108 ; BD - C3
 0118 0108 0118 0305 02C9 0108 0118     0378  5842          .WORD   ^X0118,^X0108,^X02C9,^X0305,^X0118,^X0108,^X0118 ; C4 - CA
 0108 0158 0108 0158 0108 0118 0108     0386  5843          .WORD   ^X0108,^X0118,^X0108,^X0158,^X0108,^X0158,^X0108 ; CB - D1
 0118 0265 0243 008E 0221 0108 0158     0394  5844          .WORD   ^X0158,^X0108,^X0221,^X008E,^X0243,^X0265,^X0118 ; D2 - D8
 0206 0158 0206 03A9 0398 0108 0118     03A2  5845          .WORD   ^X0118,^X0108,^X0398,^X03A9,^X0206,^X0158,^X0206 ; D9 - DF
 00FF 00DF 00DF 00DF 00DF 00BF 00BF     03B0  5846          .WORD   ^X00BF,^X00BF,^X00DF,^X00DF,^X00DF,^X00DF,^X00FF ; E0 - E6
 0108 0108 0108 0108 039E 039D 00FF     03BE  5847          .WORD   ^X00FF,^X039D,^X039E,^X0108,^X0108,^X0108,^X0108 ; E7 - ED
 03A2 0118 0118 0108 0108 0108 0108     03CC  5848          .WORD   ^X0108,^X0108,^X0108,^X0108,^X0118,^X0118,^X03A2 ; EE - F4
 0128 0128 0108 0326 0158 0158 03A2     03DA  5849          .WORD   ^X03A2,^X0158,^X0158,^X0326,^X0108,^X0128,^X0128 ; F5 - FB
                03F8 03F8 0776 0776     03E8  5850          .WORD   ^X0776,^X0776,^X03F8,^X03F8                       ; FC - FF
                                        03F0  5851
                                        03F0  5852 ;
                                        03F0  5853 ; Expected CS Addresses for Subtest 3
                                        03F0  5854 ;
                                        03F0  5855
 03F8 03F8 03F8 03F8 03F8 03F8 03F8     03F0  5856 31$:     .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 00 - 06
 0108 0108 0381 0108 03BF 03BE 03F8     03FE  5857          .WORD   ^X03F8,^X03BE,^X03BF,^X0108,^X0381,^X0108,^X0108 ; 07 - 0D
 03F8 03F8 03F8 03F8 03F8 0158 0128     040C  5858          .WORD   ^X0128,^X0158,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 0E - 14
 03F8 03F8 03F8 03F8 0396 03A4 03F8     041A  5859          .WORD   ^X03F8,^X03A4,^X0396,^X03F8,^X03F8,^X03F8,^X03F8 ; 15 - 1B
 03B9 0288 03B9 03F8 03F8 03F8 03F8     0428  5860          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03B9,^X0288,^X03B9 ; 1C - 22
 03AE 03AE 0288 03BD 0288 03BC 0288     0436  5861          .WORD   ^X0288,^X03BC,^X0288,^X03BD,^X0288,^X03AE,^X03AE ; 23 - 29
 03F8 03B2 03B2 03AF 03AF 03B4 03B4     0444  5862          .WORD   ^X03B4,^X03B4,^X03AF,^X03AF,^X03B2,^X03B2,^X03F8 ; 2A - 30
 03B9 03BA 03B7 03B7 0158 00E5 03F8     0452  5863          .WORD   ^X03F8,^X00E5,^X0158,^X03B7,^X03B7,^X03BA,^X03B9 ; 31 - 37
 0158 0108 01DE 03B5 03B5 03B6 03C1     0460  5864          .WORD   ^X03C1,^X03B6,^X03B5,^X03B5,^X01DE,^X0108,^X0158 ; 38 - 3E
 0108 0118 0108 0118 0108 0118 0206     046E  5865          .WORD   ^X0206,^X0118,^X0108,^X0118,^X0108,^X0118,^X0108 ; 3F - 45
 00E6 00E6 00E6 00E6 00E6 0108 0118     047C  5866          .WORD   ^X0118,^X0108,^X00E6,^X00E6,^X00E6,^X00E6,^X00E6 ; 46 - 4C
```

```
 0329 0158 0108 0158 0168 00E6 00E6  048A  5867          .WORD   ^X00E6,^X00E6,^X0168,^X0158,^X0108,^X0158,^X0329 ; 4D - 53
 03F8 03F8 0158 03F8 036D 0108 0108  0498  5868          .WORD   ^X0108,^X0108,^X036D,^X03F8,^X0158,^X03F8,^X03F8 ; 54 - 5A
 01B8 01C8 0158 0158 0128 0128 03F8  04A6  5869          .WORD   ^X03F8,^X0128,^X0128,^X0158,^X0158,^X01C8,^X01B8 ; 5B - 61
 00E6 01B8 01C8 01B8 01C8 01B8 01C8  04B4  5870          .WORD   ^X01C8,^X01B8,^X01C8,^X01B8,^X01C8,^X01B8,^X00E6 ; 62 - 68
 0359 01DA 01DA 01DA 00E6 00E6 00E6  04C2  5871          .WORD   ^X00E6,^X00E6,^X00E6,^X01DA,^X01DA,^X01DA,^X0359 ; 69 - 6F
 036D 0252 0108 032D 0158 01B8 0158  04D0  5872          .WORD   ^X0158,^X01B8,^X0158,^X032D,^X0108,^X0252,^X036D ; 70 - 76
 0158 022A 0178 0108 0178 0108 03F8  04DE  5873          .WORD   ^X03F8,^X0108,^X0178,^X0108,^X0178,^X022A,^X0158 ; 77 - 7D
 0118 0108 0118 0108 0118 020D 0158  04EC  5874          .WORD   ^X0158,^X020D,^X0118,^X0108,^X0118,^X0108,^X0118 ; 7E - 84
 0108 0118 0108 0118 02CD 02B9 0108  04FA  5875          .WORD   ^X0108,^X02B9,^X02CD,^X0118,^X0108,^X0118,^X0108 ; 85 - 8B
 0158 0108 0158 0108 0158 0108 0118  0508  5876          .WORD   ^X0118,^X0108,^X0158,^X0108,^X0158,^X0108,^X0158 ; 8C - 92
 00E5 00E5 0265 0243 008E 0221 0108  0516  5877          .WORD   ^X0108,^X0221,^X008E,^X0243,^X0265,^X00E5,^X00E5 ; 93 - 99
 0118 0206 0158 0108 0108 01DE 01DE  0524  5878          .WORD   ^X01DE,^X01DE,^X0108,^X0108,^X0158,^X0206,^X0118 ; 9A - A0
 0301 02BF 0108 0118 0108 0118 0108  0532  5879          .WORD   ^X0108,^X0118,^X0108,^X0118,^X0108,^X02BF,^X0301 ; A1 - A7
 0158 0108 0118 0108 0118 0108 0118  0540  5880          .WORD   ^X0118,^X0108,^X0118,^X0108,^X0118,^X0108,^X0158 ; A8 - AE
 008E 0221 0108 0158 0108 0158 0108  054E  5881          .WORD   ^X0108,^X0158,^X0108,^X0158,^X0108,^X0221,^X008E ; AF - B5
 03C6 03A6 03A7 029E 0292 0265 0243  055C  5882          .WORD   ^X0243,^X0265,^X0292,^X029E,^X03A7,^X03A6,^X03C6 ; B6 - BC
 0108 0118 0108 0118 03C9 03C8 03C7  056A  5883          .WORD   ^X03C7,^X03C8,^X03C9,^X0118,^X0108,^X0118,^X0108 ; BD - C3
 0118 0108 0118 0305 02C9 0108 0118  0578  5884          .WORD   ^X0118,^X0108,^X02C9,^X0305,^X0118,^X0108,^X0118 ; C4 - CA
 0108 0158 0108 0158 0108 0118 0108  0586  5885          .WORD   ^X0108,^X0118,^X0108,^X0158,^X0108,^X0158,^X0108 ; CB - D1
 0118 0265 0243 008E 0221 0108 0158  0594  5886          .WORD   ^X0158,^X0108,^X0221,^X008E,^X0243,^X0265,^X0118 ; D2 - D8
 0206 0158 0206 03A9 03C5 0108 0118  05A2  5887          .WORD   ^X0118,^X0108,^X03C5,^X03A9,^X0206,^X0158,^X0206 ; D9 - DF
 00FF 00DF 00DF 00DF 00DF 00BF 00BF  05B0  5888          .WORD   ^X00BF,^X00BF,^X00DF,^X00DF,^X00DF,^X00DF,^X00FF ; E0 - E6
 0108 0108 0108 0108 039E 039D 00FF  05BE  5889          .WORD   ^X00FF,^X039D,^X039E,^X0108,^X0108,^X0108,^X0108 ; E7 - ED
 03A2 0118 0118 0108 0108 0108 0108  05CC  5890          .WORD   ^X0108,^X0108,^X0108,^X0108,^X0118,^X0118,^X03A2 ; EE - F4
 0128 0128 0108 0326 0158 0158 03A2  05DA  5891          .WORD   ^X03A2,^X0158,^X0158,^X0326,^X0108,^X0128,^X0128 ; F5 - FB
                03F8 03F8 0776 0776  05E8  5892          .WORD   ^X0776,^X0776,^X03F8,^X03F8                      ; FC - FF
                                     05F0  5893
                                     05F0  5894 BEGIN_CPU_DATA
                                     0002  5895
                                     0002  5896 DCS_DATA
                                     0001  5897
                                     0001  5898 ; Micro-code to perform the instruction stream initialization and IRD1TST.
                                     0001  5899
U 00, 7700,C800,0364,0300,0470,1801  0001  5900 ;% 00    NOP                                  ; get cpu up to speed
U 01, 5700,C840,0364,0308,0470,1802  000C  5904 ;% 01    R[TEMP2]_RDM                         ; put opcode in R[TEMP2]
U 02, 7700,F800,4ECF,4FFF,8470,1803  0017  5908 ;% 02    LONLIT_[62616000]                    ; load opspecs in lonlit reg
U 03, 7700,8800,5BE5,03D4,0470,1804  0022  5912 ;% 03    Q_R[LONLIT]                          ; transfer opspecs to Q
U 04, 7700,0842,00A4,0308,0470,1805  002D  5916 ;% 04    R[TEMP2]_M[TEMP2].OR.Q               ; OR opcode with opspecs
U 05, 7700,4868,5BE4,0308,0470,1806  0038  5920 ;% 05    M[TEMP8]_R[TEMP2]                    ; save in M[TEMP8] for error
U 06, 7700,C800,5BE4,03D8,4A70,1807  0043  5924 ;% 06    VA_R[ZERO]                           ; zero the VA
U 07, 7701,C800,5BE4,0208,5590,1808  004E  5928 ;% 07    WRITE.LONG R[TEMP2]                  ; write data for XB0
U 08, 7700,0800,5BE4,0308,5470,1809  0059  5932 ;% 08    WDR_UNROT(R[TEMP2])                  ; validate cache location 0
U 09, 7700,C800,0364,0300,4470,180A  0064  5936 ;% 09    VA_VA+4                              ; update VA by 4
U 0A, 7700,3800,4CCD,4DCE,0470,180B  006F  5940 ;% 0A    LONLIT_[66656463]                    ; load opspecs in lonlit reg
U 0B, 7700,4869,5BE4,03D4,0470,180C  007A  5944 ;% 0B    M[TEMP9]_R[LONLIT]                   ; save in M[TEMP9] for error
U 0C, 7701,8800,5BE4,02D4,5590,180D  0085  5948 ;% 0C    WRITE.LONG R[LONLIT]                 ; write data for XB1
U 0D, 7700,C800,5BE4,03D4,5470,180E  0090  5952 ;% 0D    WDR_UNROT(R[LONLIT])                 ; validate cache location 4
U 0E, 7700,C800,5BE4,03D8,4870,180F  009B  5956 ;% 0E    PC_R[ZERO]                           ; zero PC to force prefetch
U 0F, 7700,C800,0364,1700,0470,1810  00A6  5960 ;% 0F    BUT/IRD1TST                          ; perform ird1
U 10, 7300,4800,0364,0300,0470,1811  00B1  5964 ;% 10    STOP.CLOCK                           ; halt
                                     00BC  5968
                                     00BC  5969 ; IRDX execution to trap generated address
                                     00BC  5970
U 11, 7700,4800,0364,0300,0470,1812  00BC  5971 ;% 11    NOP                                  ; get cpu up to speed
U 12, 7700,8800,0364,0700,0470,1813  00C7  5975 ;% 12    BUT/IRDX                             ; do irdx
```

```
U 13, 7300,4800,0364,0300,0470,1814  00D2  5979 ;% 13   STOP.CLOCK                                  ; halt
                                     00DD  5983
                                     00DD  5984 ; FPA enable control and status check
                                     00DD  5985
U 14, 7700,C800,0364,0300,0470,1815  00DD  5986 ;% 14   NOP                                         ; get cpu up to speed
U 15, 5700,C800,0364,0300,2A70,1816  00E8  5990 ;% 15   WCTRL/FPA.ENABLE_WB5,WB_RDM                 ; write FPA enable bit
U 16, 7700,4800,0364,0300,0470,1817  00F3  5994 ;% 16   NOP                                         ; ...
U 17, 7700,4800,0364,C300,0470,1800  00FE  5998 ;% 17   BUT/NO.FPA,NEXT/1800                        ; check FPA status
U 18, 7300,C800,0364,0300,0470,1819  0109  6002 ;% 18   STOP.CLOCK                                  ; halt
                                     0114  6006
                                     0114  6007 ; Control-P interrupt disable
                                     0114  6008
U 19, 7700,C800,0364,0300,0470,181A  0114  6009 ;% 19   NOP                                         ; get the cpu up to speed
U 1A, 7700,F800,7F9F,FFFF,8470,181B  011F  6013 ;% 1A   LONLIT_[00C00000]                            ; address of s&c bit in crar
U 1B, 7700,4800,5BE4,03D4,2470,181C  012A  6017 ;% 1B   WCTRL/LOADCRAR,WB_R[LONLIT]                 ; address the s&c bit
U 1C, 7700,R800,7FDF,FFFF,8470,181D  0135  6021 ;% 1C   LONLIT_[00400000]                            ; data for ctrl-p int. disable
U 1D, 7700,4800,5BE4,03D4,2C70,181E  0140  6025 ;% 1D   WCTRL/CONWRITE,WB_R[LONLIT]                 ; disable ctrl-p interrupts
U 1E, 7300,C800,0364,0300,0470,181F  014B  6029 ;% 1E   STOP.CLOCK                                  ; halt
                                     0156  6033
                                     0156  6034 ; Control-P interrupt enable
                                     0156  6035
U 1F, 7700,C800,0364,0300,0470,1820  0156  6036 ;% 1F   NOP                                         ; get the cpu up to speed
U 20, 7700,F800,7F9F,FFFF,8470,1821  0161  6040 ;% 20   LONLIT_[00C00000]                            ; address of s&c bit in crar
U 21, 7700,C800,5BE4,03D4,2470,1822  016C  6044 ;% 21   WCTRL/LOADCRAR,WB_R[LONLIT]                 ; address the s&c bit
U 22, 7700,7800,7FB7,FFFF,8470,1823  0177  6048 ;% 22   LONLIT_[00900000]                            ; data for ctrl-p int. enable
U 23, 7700,4800,5BE4,03D4,2C70,1824  0182  6052 ;% 23   WCTRL/CONWRITE,WB_R[LONLIT]                 ; enable ctrl-p interrupts
U 24, 7300,C800,0364,0300,0470,1825  018D  6056 ;% 24   STOP.CLOCK                                  ; halt
                                     0198  6060
                                     0198  6061 ; Set-up RTEMP0 and RTEMP1 for error callout
                                     0198  6062
U 25, 7700,C800,0364,0300,0470,1826  0198  6063 ;% 25   NOP                                         ; get cpu up to speed
U 26, 7700,0848,5924,0300,0470,1827  01A3  6067 ;% 26   R[TEMP0]_M[TEMP8]                            ; move XB0 data to R[TEMP0]
U 27, 7700,0849,5924,0304,0470,1828  01AE  6071 ;% 27   R[TEMP1]_M[TEMP9]                            ; move XB1 data to R[TEMP1]
U 28, 7300,C800,0364,0300,0470,1829  01B9  6075 ;% 28   STOP.CLOCK                                  ; halt
                                     01C4  6079
                                     01C4  6080 RTEMP_DATA
                                     0001  6081
                         00000000    0001  6082         .LONG   0                    ; rtemp0 = contents of xb0
                         00000000    0005  6083         .LONG   0                    ; rtemp1 = contents of xb1
                         00000000    0009  6084         .LONG   0                    ; rtemp2 = instruction
                                     000D  6085
                                     000D  6086 END_CPU_DATA
                                     05F0  6087
                                     05F0  6088 END_TEST_DATA
                                     05F0
                                     05F0       ;-----------------------------------------------------------------------
                                     05F0  6089
                                     05F0  6090 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
0031        .SBTTL  "TEST   0C7     First IRDX Instruction Address Decode Test 2
0031  6092 ;*******************************************************************************
0031  6093 ;++
0031  6094 ;
0031  6095 ; FUNCTIONAL DESCRIPTION
0031  6096 ;
0031  6097 ;       This test will examine the signal ROM OS INH H from the Native Mode
0031  6098 ;       IRDX roms when the IRD Counter is equal to zero and the operand
0031  6099 ;       specifier is for a memory reference.
0031  6100 ;
0031  6101 ; TEST ALGORITHM
0031  6102 ;
0031  6103 ;       This test consists of three subtests.  The first subtest will examine
0031  6104 ;       the IRDX roms when the Floating Point Accelerator (FPA) is not present
0031  6105 ;       in the system.  The second subtest will examine the IRDX roms when the
0031  6106 ;       FPA is present but disabled.  The third subtest will examine the IRDX
0031  6107 ;       roms when the FPA is present and enabled.  The algorithm is as follows:
0031  6108 ;
0031  6109 ;        1.  Initialize the processor.
0031  6110 ;        2.  Disable CTRL/P interrupts.
0031  6111 ;        3.  Check to see if FPA present
0031  6112 ;               If FPA present
0031  6113 ;                   Then skip to Subtest 2
0031  6114 ;               Else fall through to Subtest 1
0031  6115 ;
0031  6116 ;       Subtest 1
0031  6117 ;
0031  6118 ;        4.  Set-up flag to expect a micro trap.
0031  6119 ;        5.  Initialize loop I for 256 opcodes (00-FF).
0031  6120 ;        6.  Store the index I in a location that will be used to load into the
0031  6121 ;            WDREG.  The value is used as the opcode.
0031  6122 ;        7.  Loop on error from here.
0031  6123 ;        8.  Execute the following test microcode:
0031  6124 ;               a.  NOP to get the RDM up to speed.
0031  6125 ;               b.  Read the WDREG into RTEMP2 (opcode).
0031  6126 ;               c.  Load the operands into the LONLIT register.
0031  6127 ;               d.  Store the operands into the Q register to prepare for ORing
0031  6128 ;                   the opcode with the operands.
0031  6129 ;               e.  OR the opcode and operands together (XB0 data).
0031  6130 ;               f.  Store XB0 data in M[TEMP8] in case needed for error callout.
0031  6131 ;               g.  Zero the VA.
0031  6132 ;               h.  Validate cache location 0 by first writing the data in
0031  6133 ;                   RTEMP2 then reading the data back.
0031  6134 ;               i.  Update the VA by 4.
0031  6135 ;               j.  Load the second operand data into the LONLIT register
0031  6136 ;                   (XB1 data).
0031  6137 ;               k.  Store XB1 data in M[TEMP9] in case needed for error callout.
0031  6138 ;               l.  Validate cache location 4 by first writing the data in
0031  6139 ;                   the LONLIT register then reading the data back.
0031  6140 ;               m.  Load the PC with zero to cause a prefetch to take place.
0031  6141 ;               n.  Do an IRD1TST to init the IRD counter and prepare for
0031  6142 ;                   instruction/operand specifier evaluation.
0031  6143 ;               o.  Halt.
0031  6144 ;        9.  Perform the following test microcode.
0031  6145 ;               a.  NOP to get the RDM up to speed.
```

```
0031  6146 ;                  b.  Do an IRDX and strobe the VBUS.
0031  6147 ;                  c.  Halt.
0031  6148 ;          10.  Compare the VBUS data with the expected value and bit position
0031  6149 ;               from the VBUS Table.
0031  6150 ;          11.  If the two do not equal, execute the following microcode and
0031  6151 ;               signal an error (this is done to set-up for printing RTEMP0 & 1
0031  6152 ;               on the error callout):
0031  6153 ;                  a.  NOP to get the RDM up to speed.
0031  6154 ;                  b.  Load RTEMP0 with the contents of M[TEMP8]
0031  6155 ;                  c.  Load RTEMP1 with the contents of M[TEMP9]
0031  6156 ;                  d.  Halt
0031  6157 ;          12.  Continue with loop I for the 256 VAX opcodes.
0031  6158 ;          13.  Re-enable CTRL/P interrupts then skip to next tests.
0031  6159 ;
0031  6160 ;          Subtest 2
0031  6161 ;
0031  6162 ;          14.  Load the FPA disable bit (bit <5> equal to 0).
0031  6163 ;          15.  Loop on error from here.
0031  6164 ;          16.  Set-up to expect micro trap.
0031  6165 ;          17.  Execute the test microcode.  This is done only to get the FPA to
0031  6166 ;               a state where it can be disabled or enabled.  If this had not been
0031  6167 ;               done before then the FPA would be setting in the initialize state
0031  6168 ;               which forces FPA enable.
0031  6169 ;          18.  Execute the following microcode to disable the FPA:
0031  6170 ;                  a.  NOP to get the RDM up to speed.
0031  6171 ;                  b.  Read the RDM WDREG onto the W Bus and then write the W Bus
0031  6172 ;                      to the FPA.
0031  6173 ;                  c.  Perform a NO.FPA test to insure that the FPA got disabled.
0031  6174 ;                  d.  Halt.
0031  6175 ;          19.  Check the Trap register to insure that the FPA got disabled.
0031  6176 ;          20.  If the FPA is not disabled, then signal an error.
0031  6177 ;          21.  Set-up flag to expect a micro trap.
0031  6178 ;          22.  Initialize loop I for 256 opcodes (00-FF).
0031  6179 ;          23.  Store the index I in a location that will be used to load into the
0031  6180 ;               WDREG.  The value is used as the opcode.
0031  6181 ;          24.  Loop on error from here.
0031  6182 ;          25.  Execute the following test microcode:
0031  6183 ;                  a.  NOP to get the RDM up to speed.
0031  6184 ;                  b.  Read the WDREG into RTEMP2 (opcode).
0031  6185 ;                  c.  Load the operands into the LONLIT register.
0031  6186 ;                  d.  Store the operands into the Q register to prepare for ORing
0031  6187 ;                      the opcode with the operands.
0031  6188 ;                  e.  OR the opcode and operands together (XB0 data).
0031  6189 ;                  f.  Store XB0 data in M[TEMP8] in case needed for error callout.
0031  6190 ;                  g.  Zero the VA.
0031  6191 ;                  h.  Validate cache location 0 by first writing the data in
0031  6192 ;                      RTEMP2 then reading the data back.
0031  6193 ;                  i.  Update the VA by 4.
0031  6194 ;                  j.  Load the second operand data into the LONLIT register
0031  6195 ;                      (XB1 data).
0031  6196 ;                  k.  Store XB1 data in M[TEMP9] in case needed for error callout.
0031  6197 ;                  l.  Validate cache location 4 by first writing the data in
0031  6198 ;                      the LONLIT register then reading the data back.
0031  6199 ;                  m.  Load the PC with zero to cause a prefetch to take place.
0031  6200 ;                  n.  Do an IRD1 to init the IRD counter and prepare for instruc-
```

```
0031  6201 ;                    tion/operand specifier evaluation.
0031  6202 ;                 o.  Halt.
0031  6203 ;       26.  Perform the following test microcode.
0031  6204 ;                 a.  NOP to get the RDM up to speed.
0031  6205 ;                 b.  Do an IRDX and strobe the VBUS.
0031  6206 ;                 c.  Halt.
0031  6207 ;       27.  Compare the VBUS data with the expected value and bit position
0031  6208 ;            from the VBUS Table.
0031  6209 ;       28.  If the two do not equal, execute the following microcode and
0031  6210 ;            signal an error (this is done to set-up for printing RTEMP0 & 1
0031  6211 ;            on the error callout):
0031  6212 ;                 a.  NOP to get the RDM up to speed.
0031  6213 ;                 b.  Load RTEMP0 with the contents of M[TEMP8]
0031  6214 ;                 c.  Load RTEMP1 with the contents of M[TEMP9]
0031  6215 ;                 d.  Halt
0031  6216 ;       29.  Continue with loop I for the 256 VAX opcodes.
0031  6217 ;       30.  Go to Subtest 3.
0031  6218 ;
0031  6219 ;       Subtest 3
0031  6220 ;
0031  6221 ;       31.  Load the FPA enable bit (bit <5> equal to 1).
0031  6222 ;       32.  Loop on error from here.
0031  6223 ;       33.  Set-up to expect micro trap.
0031  6224 ;       34.  Execute the following microcode to enable the FPA:
0031  6225 ;                 a.  NOP to get the RDM up to speed.
0031  6226 ;                 b.  Read the RDM WDREG onto the W Bus and then write the W Bus
0031  6227 ;                     to the FPA.
0031  6228 ;                 c.  Perform a NO.FPA test to insure that the FPA got enabled.
0031  6229 ;                 d.  Halt.
0031  6230 ;       35.  Check the Trap register to insure that the FPA got enabled.
0031  6231 ;       36.  If the FPA is not enabled, then signal an error.
0031  6232 ;       37.  Set-up flag to expect a micro trap.
0031  6233 ;       38.  Initialize loop I for 256 opcodes (00-FF).
0031  6234 ;       39.  Store the index I in a location that will be used to load into the
0031  6235 ;            WDREG.  The value is used as the opcode.
0031  6236 ;       40.  Loop on error from here.
0031  6237 ;       41.  Set then Clear the DCLO bit in the RDCTRL register to perform an
0031  6238 ;            initialize.  This is done because when the test microcode
0031  6239 ;            executes, if the opcode was an FP opcode then there is a
0031  6240 ;            possibility that the processor could be setting there in an
0031  6241 ;            FPAWAIT state (M clock and D clock stalled).  If this were to
0031  6242 ;            happen then the cpu clocks would no longer run and the remaining
0031  6243 ;            test opcodes would fail.
0031  6244 ;       42.  Execute the following test microcode:
0031  6245 ;                 a.  NOP to get the RDM up to speed.
0031  6246 ;                 b.  Read the WDREG into RTEMP2 (opcode).
0031  6247 ;                 c.  Load the operands into the LONLIT register.
0031  6248 ;                 d.  Store the operands into the Q register to prepare for ORing
0031  6249 ;                     the opcode with the operands.
0031  6250 ;                 e.  OR the opcode and operands together (XB0 data).
0031  6251 ;                 f.  Store XB0 data in M[TEMP8] in case needed for error callout.
0031  6252 ;                 g.  Zero the VA.
0031  6253 ;                 h.  Validate cache location 0 by first writing the data in
0031  6254 ;                     RTEMP2 then reading the data back.
0031  6255 ;                 i.  Update the VA by 4.
```

```
0031  6256 ;                  j.  Load the second operand data into the LONLIT register
0031  6257 ;                      (XB1 data).
0031  6258 ;                  k.  Store XB1 data in M[TEMP9] in case needed for error callout.
0031  6259 ;                  l.  Validate cache location 4 by first writing the data in
0031  6260 ;                      the LONLIT register then reading the data back.
0031  6261 ;                  m.  Load the PC with zero to cause a prefetch to take place.
0031  6262 ;                  n.  Do an IRD1 to init the IRD counter and prepare for instruc-
0031  6263 ;                      tion/operand specifier evaluation.
0031  6264 ;                  o.  Halt.
0031  6265 ;          43.  Perform the following test microcode.
0031  6266 ;                  a.  NOP to get the RDM up to speed.
0031  6267 ;                  b.  Do an IRDX and strobe the VBUS.
0031  6268 ;                  c.  Halt.
0031  6269 ;          44.  Compare the VBUS data with the expected value and bit position
0031  6270 ;               from the VBUS Table.
0031  6271 ;          45.  If the two do not equal, execute the following microcode and
0031  6272 ;               signal an error (this is done to set-up for printing RTEMP0 & 1
0031  6273 ;               on the error callout):
0031  6274 ;                  a.  NOP to get the RDM up to speed.
0031  6275 ;                  b.  Load RTEMP0 with the contents of M[TEMP8]
0031  6276 ;                  c.  Load RTEMP1 with the contents of M[TEMP9]
0031  6277 ;                  d.  Halt
0031  6278 ;          46.  Continue with loop I for the 256 VAX opcodes.
0031  6279 ;          47.  Re-enable CTRL/P interrupts.
0031  6280 ;          48.  Skip to next tests.
0031  6281 ;
0031  6282 ; TEST PATTERNS
0031  6283 ;
0031  6284 ;       For this test the opcodes are generated by a simple count pattern.  The
0031  6285 ;       opcodes are based on the 256 base machine instructions as specified in
0031  6286 ;       the VAX architecture handbook.  The generated opcodes are written to
0031  6287 ;       the CPU and combined (ORed) with three bytes of operand data to form
0031  6288 ;       a machine instruction.  The operands all specify Register Deferred mode
0031  6289 ;       using registers 0-6 and are loaded from the LONLIT register.
0031  6290 ;
0031  6291 ;       Also required are two tables of expected VBUS values for the first IRDX
0031  6292 ;       cycle.  One table contains the VBUS values for the first IRDX when the
0031  6293 ;       FPA is disabled, the other for when it is enabled.
0031  6294 ;
0031  6295 ; LOGIC DESCRIPTION
0031  6296 ;
0031  6297 ;       This test will check the validity of the IRDX roms on the DPM module.
0031  6298 ;       The IRDX roms are addressed by bits <7:0> of the Instruction Register.
0031  6299 ;       Along with being addressed by the IR, there are three other signals
0031  6300 ;       that affect the address.  They are IRD CTR 0 H, FPA ENABLED L, and
0031  6301 ;       REG MODE H.  This test will examine every 8th(hex) address starting at
0031  6302 ;       address 2 with FPA disabled and starting at address 0 with FPA enabled.
0031  6303 ;
0031  6304 ; ASSUMPTIONS
0031  6305 ;
0031  6306 ;       This test assumes the following logic has already been tested; the
0031  6307 ;       CPU's data path, the cache, the XB, and the IR and OSR.
0031  6308 ;
0031  6309 ; ERROR DESCRIPTION
0031  6310 ;
```

```
                        0031  6311 ;       Subtest 1
                        0031  6312 ;
                        0031  6313 ;               Expected CS Address
                        0031  6314 ;               Received CS Address
                        0031  6315 ;
                        0031  6316 ;       Subtest 2
                        0031  6317 ;
                        0031  6318 ;               First IFERROR
                        0031  6319 ;
                        0031  6320 ;                       Expected CS Address
                        0031  6321 ;                       Received CS Address
                        0031  6322 ;
                        0031  6323 ;               Second IFERROR
                        0031  6324 ;
                        0031  6325 ;                       Expected VBUS ( 00000ynn )      y  = VBUS bit value
                        0031  6326 ;                       Received VBUS ( 00000ynn )      nn = VBUS bit number
                        0031  6327 ;                       Loop I count
                        0031  6328 ;                       Contents of XB0
                        0031  6329 ;                       Contents of XB1
                        0031  6330 ;
                        0031  6331 ;       Subtest 3
                        0031  6332 ;
                        0031  6333 ;               First IFERROR
                        0031  6334 ;
                        0031  6335 ;                       Expected CS Address
                        0031  6336 ;                       Received CS Address
                        0031  6337 ;
                        0031  6338 ;               Second IFERROR
                        0031  6339 ;
                        0031  6340 ;                       Expected VBUS ( 00000ynn )      y  = VBUS bit value
                        0031  6341 ;                       Received VBUS ( 00000ynn )      nn = VBUS bit number
                        0031  6342 ;                       Loop I count
                        0031  6343 ;                       Contents of XB0
                        0031  6344 ;                       Contents of XB1
                        0031  6345 ;--
                        0031  6346 ;*******************************************************************************
                        0031  6347
                        0031  6348         INITIALIZE                                    ; initialize the comet cpu
                        0033  6349         FETCH           <^X19>                        ; disable ctrl-p interrupts
                        0038  6350         BRSTCLK         <^X1E>                        ; ...
                        003C  6351         FETCH           <^X16>                        ; execute a but/no.fpa to check
                        0041  6352         BRSTCLK         <^X18>                        ; for fpa present
                        0045  6353         CMPREGMSK       CSTRP,1$,,2$,,W               ; is the fpa present?
                        0051  6354         SKIP            100$,ONERROR                  ; if so then skip to subtest 2
                        0058  6355
                        0058  6356         SUBTEST         1
                        005B
                              005B      ;////////////////////////////////////////////////////////////////////////////////
                        005B  6357 ;+
                        005B  6358 ;
                        005B  6359 ; FPA is not present.
                        005B  6360 ;
                        005B  6361 ;-
                        005B  6362         EXPUTRAP                                      ; else expect non-dcs addresses
                        005D  6363         LOOP            I,1,256                       ; loop for 256 opcodes
```

```
0064  6364          SAVEINDEX       I,9$                    ; save the loop count
0069  6365          LOADREG         WDREG,9$,,L             ; load wdreg with loop count
0071  6366          ERRLOOP                                 ; Loop on error from here
0073  6367          FETCH           0                       ; load the istrm data and do an
0078  6368          BRSTCLK         <^X10>,INH              ; ird1tst
007C  6369          FETCH           <^X11>                  ; execute the irdx
0081  6370          BRSTCLK         <^X13>,INH              ; ...
0085  6371          CMPVBUS         30$,I                   ; Check VBUS bit for error
008A  6372          SKIP            50$,NOERROR             ; continue testing if no error
0091  6373          FETCH           <^X25>                  ; start dcs program
0096  6374          BRSTCLK         <^X28>                  ; end dcs program
009A  6375          IFERROR         2,<<IRD>>,<<MIC>>       ; if bad address, signal error
00A1  6376 50$:     ENDLOOP         I                       ; continue loop
00A4  6377          FETCH           <^X1F>                  ; re-enable ctrl-p interrupts
00A9  6378          BRSTCLK         <^X24>                  ; ...
00AD  6379          SKIP                                    ; skip on to the next test
00B4  6380
00B4  6381 100$:    SUBTEST         2
00B7
00B7          ;////////////////////////////////////////////////////////////////////////////////
00B7  6382 ;+
00B7  6383 ;
00B7  6384 ; FPA present but not enabled
00B7  6385 ;
00B7  6386 ;-
00B7  6387          LOADREG         WDREG,7$,,L             ; load fpa disable bit
00BF  6388          ERRLOOP                                 ; loop on error here
00C1  6389          EXPUTRAP                                ; expect micro trap
00C3  6390          FETCH           0                       ; start dcs program
00C8  6391          BRSTCLK         <^X10>,INH              ; end dcs program
00CC  6392          FETCH           <^X14>                  ; execute the disable
00D1  6393          BRSTCLK         <^X18>,INH              ; and read back fpa status
00D5  6394          CMPREGMSK       CSTRP,3$,,5$,,W         ; is the fpa disabled?
00E1  6395          IFERROR         ,,<FPA>                 ; if not, signal an error
00E7  6396          EXPUTRAP                                ; else expect non-dcs addresses
00E9  6397          LOOP            I,1,256                 ; loop for 256 opcodes
00F0  6398          SAVEINDEX       I,9$                    ; save the loop count
00F5  6399          LOADREG         WDREG,9$,,L             ; load wdreg with loop count
00FD  6400          ERRLOOP                                 ; Loop on error from here
00FF  6401          FETCH           0                       ; execute a but/ird1tst
0104  6402          BRSTCLK         <^X10>,INH              ; ...
0108  6403          FETCH           <^X11>                  ; execute an irdx
010D  6404          BRSTCLK         <^X13>,INH              ; ...
0111  6405          CMPVBUS         30$,I                   ; check VBUS bit for error
0116  6406          SKIP            51$,NOERROR             ; continue testing if no error
011D  6407          FETCH           <^X25>                  ; start dcs program
0122  6408          BRSTCLK         <^X28>                  ; end dcs program
0126  6409          IFERROR         2,<<IRD>>,<<MIC>>       ; if error then signal an error
012D  6410 51$:     ENDLOOP         I                       ; continue loop
0130  6411
0130  6412          SUBTEST         3
0133
0133          ;////////////////////////////////////////////////////////////////////////////////
0133  6413 ;+
0133  6414 ;
```

```
                        0133  6415 ; FPA present and enabled
                        0133  6416 ;
                        0133  6417 ;-
                        0133  6418          LOADREG         WDREG,8$,,L             ; load fpa enable bit
                        013B  6419          ERRLOOP                                 ; loop on error from here
                        013D  6420          FETCH           <^X14>                  ; enable the fpa
                        0142  6421          BRSTCLK         <^X18>,INH              ; and read back fpa enable
                        0146  6422          CMPREGMSK       CSTRP,4$,,5$,,W         ; is the fpa enabled?
                        0152  6423          IFERROR         ,,<FPA>                 ; if not, signal an error
                        0158  6424          EXPUTRAP                                ; else expect non-dcs addresses
                        015A  6425          LOOP            I,1,256                 ; loop for 256 opcodes
                        0161  6426          SAVEINDEX       I,9$                    ; save the loop count
                        0166  6427          LOADREG         WDREG,9$,,L             ; load wdreg with loop count
                        016E  6428          ERRLOOP                                 ; Loop on error from here
                        0170  6429          LOADREG         RDCTRL,10$,,B           ; assert dclo
                        0178  6430          LOADREG         RDCTRL,11$,,B           ; de-assert dclo
                        0180  6431          FETCH           0                       ; execute a but/ird1tst
                        0185  6432          BRSTCLK         <^X10>,INH              ; ...
                        0189  6433          FETCH           <^X11>                  ; execute the but/irdx
                        018E  6434          BRSTCLK         <^X13>,INH              ; ...
                        0192  6435          CMPVBUS         31$,I                   ; check VBUS bit for error
                        0197  6436          SKIP            52$,NOERROR             ; continue testing if no error
                        019E  6437          FETCH           <^X25>                  ; start dcs program
                        01A3  6438          BRSTCLK         <^X28>                  ; end dcs program
                        01A7  6439          IFERROR         2,<<IRD>>,<<MIC>>       ; if bad address, signal error
                        01AE  6440 52$:     ENDLOOP         I                       ; continue loop
                        01B1  6441          FETCH           <^X1F>                  ; re-enable ctrl-p interrupts
                        01B6  6442          BRSTCLK         <^X24>                  ; ...
                        01BA  6443          SKIP                                    ; end of test
                        01C1  6444
                        01C1  6445 BEGIN_TEST_DATA
                        01C1
                        01C1          ;-----------------------------------------------------------------------
                        01C1  6446
                1802    01C1  6447 1$:      .WORD   ^X1802
                3FFE    01C3  6448 2$:      .WORD   ^X3FFE
                        01C5  6449
                1801    01C5  6450 3$:      .WORD   ^X1801
                1800    01C7  6451 4$:      .WORD   ^X1800
                3FFD    01C9  6452 5$:      .WORD   ^X3FFD
                        01CB  6453
                3FFF    01CB  6454 6$:      .WORD   ^X3FFF
                        01CD  6455
            00000000    01CD  6456 7$:      .LONG   0
            00000020    01D1  6457 8$:      .LONG   ^X00000020
                        01D5  6458
            00000000    01D5  6459 9$:      .LONG   0
                  40    01D9  6460 10$:     .BYTE   DC_LOW
                  00    01DA  6461 11$:     .BYTE   ^X00
                        01DB  6462
                        01DB  6463
                        01DB  6464 ;
                        01DB  6465 ; Expected VBUS data for Subtests 1 and 2
                        01DB  6466 ;
                        01DB  6467 ; The following is a table of all the expected VBUS values for the first IRDX.
```

```
                    01DB  6468 ; A 1 means the ROM OS INH H signal is asserted and a 0 means it is not
                    01DB  6469 ; asserted.
                    01DB  6470 ;
                    01DB  6471
                01  01DB  6472 30$:    .BYTE   ^X01
                    01DC  6473          VBUSDATA        <^X1E>,1                ;00
                01  01DD  6474          .BYTE   ^X01
                    01DE  6475          VBUSDATA        <^X1E>,1                ;01
                01  01DF  6476          .BYTE   ^X01
                    01E0  6477          VBUSDATA        <^X1E>,1                ;02
                01  01E1  6478          .BYTE   ^X01
                    01E2  6479          VBUSDATA        <^X1E>,1                ;03
                01  01E3  6480          .BYTE   ^X01
                    01E4  6481          VBUSDATA        <^X1E>,1                ;04
                01  01E5  6482          .BYTE   ^X01
                    01E6  6483          VBUSDATA        <^X1E>,1                ;05
                01  01E7  6484          .BYTE   ^X01
                    01E8  6485          VBUSDATA        <^X1E>,1                ;06
                01  01E9  6486          .BYTE   ^X01
                    01EA  6487          VBUSDATA        <^X1E>,1                ;07
                01  01EB  6488          .BYTE   ^X01
                    01EC  6489          VBUSDATA        <^X1E>,1                ;08
                01  01ED  6490          .BYTE   ^X01
                    01EE  6491          VBUSDATA        <^X1E>,1                ;09
                01  01EF  6492          .BYTE   ^X01
                    01F0  6493          VBUSDATA        <^X1E>,0                ;0A
                01  01F1  6494          .BYTE   ^X01
                    01F2  6495          VBUSDATA        <^X1E>,1                ;0B
                01  01F3  6496          .BYTE   ^X01
                    01F4  6497          VBUSDATA        <^X1E>,0                ;0C
                01  01F5  6498          .BYTE   ^X01
                    01F6  6499          VBUSDATA        <^X1E>,0                ;0D
                01  01F7  6500          .BYTE   ^X01
                    01F8  6501          VBUSDATA        <^X1E>,0                ;0E
                01  01F9  6502          .BYTE   ^X01
                    01FA  6503          VBUSDATA        <^X1E>,0                ;0F
                01  01FB  6504          .BYTE   ^X01
                    01FC  6505          VBUSDATA        <^X1E>,1                ;10
                01  01FD  6506          .BYTE   ^X01
                    01FE  6507          VBUSDATA        <^X1E>,1                ;11
                01  01FF  6508          .BYTE   ^X01
                    0200  6509          VBUSDATA        <^X1E>,1                ;12
                01  0201  6510          .BYTE   ^X01
                    0202  6511          VBUSDATA        <^X1E>,1                ;13
                01  0203  6512          .BYTE   ^X01
                    0204  6513          VBUSDATA        <^X1E>,1                ;14
                01  0205  6514          .BYTE   ^X01
                    0206  6515          VBUSDATA        <^X1E>,1                ;15
                01  0207  6516          .BYTE   ^X01
                    0208  6517          VBUSDATA        <^X1E>,1                ;16
                01  0209  6518          .BYTE   ^X01
                    020A  6519          VBUSDATA        <^X1E>,1                ;17
                01  020B  6520          .BYTE   ^X01
                    020C  6521          VBUSDATA        <^X1E>,1                ;18
                01  020D  6522          .BYTE   ^X01
```

```
                020E  6523          VBUSDATA        <^X1E>,1                ;19
          01    020F  6524          .BYTE   ^X01
                0210  6525          VBUSDATA        <^X1E>,1                ;1A
          01    0211  6526          .BYTE   ^X01
                0212  6527          VBUSDATA        <^X1E>,1                ;1B
          01    0213  6528          .BYTE   ^X01
                0214  6529          VBUSDATA        <^X1E>,1                ;1C
          01    0215  6530          .BYTE   ^X01
                0216  6531          VBUSDATA        <^X1E>,1                ;1D
          01    0217  6532          .BYTE   ^X01
                0218  6533          VBUSDATA        <^X1E>,1                ;1E
          01    0219  6534          .BYTE   ^X01
                021A  6535          VBUSDATA        <^X1E>,1                ;1F
          01    021B  6536          .BYTE   ^X01
                021C  6537          VBUSDATA        <^X1E>,1                ;20
          01    021D  6538          .BYTE   ^X01
                021E  6539          VBUSDATA        <^X1E>,1                ;21
          01    021F  6540          .BYTE   ^X01
                0220  6541          VBUSDATA        <^X1E>,1                ;22
          01    0221  6542          .BYTE   ^X01
                0222  6543          VBUSDATA        <^X1E>,1                ;23
          01    0223  6544          .BYTE   ^X01
                0224  6545          VBUSDATA        <^X1E>,1                ;24
          01    0225  6546          .BYTE   ^X01
                0226  6547          VBUSDATA        <^X1E>,1                ;25
          01    0227  6548          .BYTE   ^X01
                0228  6549          VBUSDATA        <^X1E>,1                ;26
          01    0229  6550          .BYTE   ^X01
                022A  6551          VBUSDATA        <^X1E>,1                ;27
          01    022B  6552          .BYTE   ^X01
                022C  6553          VBUSDATA        <^X1E>,1                ;28
          01    022D  6554          .BYTE   ^X01
                022E  6555          VBUSDATA        <^X1E>,1                ;29
          01    022F  6556          .BYTE   ^X01
                0230  6557          VBUSDATA        <^X1E>,1                ;2A
          01    0231  6558          .BYTE   ^X01
                0232  6559          VBUSDATA        <^X1E>,1                ;2B
          01    0233  6560          .BYTE   ^X01
                0234  6561          VBUSDATA        <^X1E>,1                ;2C
          01    0235  6562          .BYTE   ^X01
                0236  6563          VBUSDATA        <^X1E>,1                ;2D
          01    0237  6564          .BYTE   ^X01
                0238  6565          VBUSDATA        <^X1E>,1                ;2E
          01    0239  6566          .BYTE   ^X01
                023A  6567          VBUSDATA        <^X1E>,1                ;2F
          01    023B  6568          .BYTE   ^X01
                023C  6569          VBUSDATA        <^X1E>,1                ;30
          01    023D  6570          .BYTE   ^X01
                023E  6571          VBUSDATA        <^X1E>,1                ;31
          01    023F  6572          .BYTE   ^X01
                0240  6573          VBUSDATA        <^X1E>,1                ;32
          01    0241  6574          .BYTE   ^X01
                0242  6575          VBUSDATA        <^X1E>,0                ;33
          01    0243  6576          .BYTE   ^X01
                0244  6577          VBUSDATA        <^X1E>,1                ;34
```

```
01  0245  6578          .BYTE   ^X01
    0246  6579          VBUSDATA        <^X1E>,1                ;35
01  0247  6580          .BYTE   ^X01
    0248  6581          VBUSDATA        <^X1E>,1                ;36
01  0249  6582          .BYTE   ^X01
    024A  6583          VBUSDATA        <^X1E>,1                ;37
01  024B  6584          .BYTE   ^X01
    024C  6585          VBUSDATA        <^X1E>,1                ;38
01  024D  6586          .BYTE   ^X01
    024E  6587          VBUSDATA        <^X1E>,1                ;39
01  024F  6588          .BYTE   ^X01
    0250  6589          VBUSDATA        <^X1E>,1                ;3A
01  0251  6590          .BYTE   ^X01
    0252  6591          VBUSDATA        <^X1E>,1                ;3B
01  0253  6592          .BYTE   ^X01
    0254  6593          VBUSDATA        <^X1E>,1                ;3C
01  0255  6594          .BYTE   ^X01
    0256  6595          VBUSDATA        <^X1E>,0                ;3D
01  0257  6596          .BYTE   ^X01
    0258  6597          VBUSDATA        <^X1E>,0                ;3E
01  0259  6598          .BYTE   ^X01
    025A  6599          VBUSDATA        <^X1E>,1                ;3F
01  025B  6600          .BYTE   ^X01
    025C  6601          VBUSDATA        <^X1E>,0                ;40
01  025D  6602          .BYTE   ^X01
    025E  6603          VBUSDATA        <^X1E>,0                ;41
01  025F  6604          .BYTE   ^X01
    0260  6605          VBUSDATA        <^X1E>,1                ;42
01  0261  6606          .BYTE   ^X01
    0262  6607          VBUSDATA        <^X1E>,1                ;43
01  0263  6608          .BYTE   ^X01
    0264  6609          VBUSDATA        <^X1E>,0                ;44
01  0265  6610          .BYTE   ^X01
    0266  6611          VBUSDATA        <^X1E>,0                ;45
01  0267  6612          .BYTE   ^X01
    0268  6613          VBUSDATA        <^X1E>,0                ;46
01  0269  6614          .BYTE   ^X01
    026A  6615          VBUSDATA        <^X1E>,0                ;47
01  026B  6616          .BYTE   ^X01
    026C  6617          VBUSDATA        <^X1E>,0                ;48
01  026D  6618          .BYTE   ^X01
    026E  6619          VBUSDATA        <^X1E>,0                ;49
01  026F  6620          .BYTE   ^X01
    0270  6621          VBUSDATA        <^X1E>,0                ;4A
01  0271  6622          .BYTE   ^X01
    0272  6623          VBUSDATA        <^X1E>,0                ;4B
01  0273  6624          .BYTE   ^X01
    0274  6625          VBUSDATA        <^X1E>,1                ;4C
01  0275  6626          .BYTE   ^X01
    0276  6627          VBUSDATA        <^X1E>,1                ;4D
01  0277  6628          .BYTE   ^X01
    0278  6629          VBUSDATA        <^X1E>,0                ;4E
01  0279  6630          .BYTE   ^X01
    027A  6631          VBUSDATA        <^X1E>,0                ;4F
01  027B  6632          .BYTE   ^X01
```

```
                027C  6633          VBUSDATA        <^X1E>,0                ;50
          01    027D  6634          .BYTE   ^X01
                027E  6635          VBUSDATA        <^X1E>,0                ;51
          01    027F  6636          .BYTE   ^X01
                0280  6637          VBUSDATA        <^X1E>,0                ;52
          01    0281  6638          .BYTE   ^X01
                0282  6639          VBUSDATA        <^X1E>,1                ;53
          01    0283  6640          .BYTE   ^X01
                0284  6641          VBUSDATA        <^X1E>,0                ;54
          01    0285  6642          .BYTE   ^X01
                0286  6643          VBUSDATA        <^X1E>,0                ;55
          01    0287  6644          .BYTE   ^X01
                0288  6645          VBUSDATA        <^X1E>,0                ;56
          01    0289  6646          .BYTE   ^X01
                028A  6647          VBUSDATA        <^X1E>,1                ;57
          01    028B  6648          .BYTE   ^X01
                028C  6649          VBUSDATA        <^X1E>,0                ;58
          01    028D  6650          .BYTE   ^X01
                028E  6651          VBUSDATA        <^X1E>,1                ;59
          01    028F  6652          .BYTE   ^X01
                0290  6653          VBUSDATA        <^X1E>,1                ;5A
          01    0291  6654          .BYTE   ^X01
                0292  6655          VBUSDATA        <^X1E>,1                ;5B
          01    0293  6656          .BYTE   ^X01
                0294  6657          VBUSDATA        <^X1E>,0                ;5C
          01    0295  6658          .BYTE   ^X01
                0296  6659          VBUSDATA        <^X1E>,0                ;5D
          01    0297  6660          .BYTE   ^X01
                0298  6661          VBUSDATA        <^X1E>,0                ;5E
          01    0299  6662          .BYTE   ^X01
                029A  6663          VBUSDATA        <^X1E>,0                ;5F
          01    029B  6664          .BYTE   ^X01
                029C  6665          VBUSDATA        <^X1E>,1                ;60
          01    029D  6666          .BYTE   ^X01
                029E  6667          VBUSDATA        <^X1E>,1                ;61
          01    029F  6668          .BYTE   ^X01
                02A0  6669          VBUSDATA        <^X1E>,1                ;62
          01    02A1  6670          .BYTE   ^X01
                02A2  6671          VBUSDATA        <^X1E>,1                ;63
          01    02A3  6672          .BYTE   ^X01
                02A4  6673          VBUSDATA        <^X1E>,1                ;64
          01    02A5  6674          .BYTE   ^X01
                02A6  6675          VBUSDATA        <^X1E>,1                ;65
          01    02A7  6676          .BYTE   ^X01
                02A8  6677          VBUSDATA        <^X1E>,1                ;66
          01    02A9  6678          .BYTE   ^X01
                02AA  6679          VBUSDATA        <^X1E>,1                ;67
          01    02AB  6680          .BYTE   ^X01
                02AC  6681          VBUSDATA        <^X1E>,1                ;68
          01    02AD  6682          .BYTE   ^X01
                02AE  6683          VBUSDATA        <^X1E>,1                ;69
          01    02AF  6684          .BYTE   ^X01
                02B0  6685          VBUSDATA        <^X1E>,1                ;6A
          01    02B1  6686          .BYTE   ^X01
                02B2  6687          VBUSDATA        <^X1E>,1                ;6B
```

```
01  02B3  6688          .BYTE   ^X01
    02B4  6689          VBUSDATA        <^X1E>,1                ;6C
01  02B5  6690          .BYTE   ^X01
    02B6  6691          VBUSDATA        <^X1E>,1                ;6D
01  02B7  6692          .BYTE   ^X01
    02B8  6693          VBUSDATA        <^X1E>,1                ;6E
01  02B9  6694          .BYTE   ^X01
    02BA  6695          VBUSDATA        <^X1E>,1                ;6F
01  02BB  6696          .BYTE   ^X01
    02BC  6697          VBUSDATA        <^X1E>,0                ;70
01  02BD  6698          .BYTE   ^X01
    02BE  6699          VBUSDATA        <^X1E>,1                ;71
01  02BF  6700          .BYTE   ^X01
    02C0  6701          VBUSDATA        <^X1E>,0                ;72
01  02C1  6702          .BYTE   ^X01
    02C2  6703          VBUSDATA        <^X1E>,1                ;73
01  02C3  6704          .BYTE   ^X01
    02C4  6705          VBUSDATA        <^X1E>,1                ;74
01  02C5  6706          .BYTE   ^X01
    02C6  6707          VBUSDATA        <^X1E>,1                ;75
01  02C7  6708          .BYTE   ^X01
    02C8  6709          VBUSDATA        <^X1E>,0                ;76
01  02C9  6710          .BYTE   ^X01
    02CA  6711          VBUSDATA        <^X1E>,1                ;77
01  02CB  6712          .BYTE   ^X01
    02CC  6713          VBUSDATA        <^X1E>,0                ;78
01  02CD  6714          .BYTE   ^X01
    02CE  6715          VBUSDATA        <^X1E>,0                ;79
01  02CF  6716          .BYTE   ^X01
    02D0  6717          VBUSDATA        <^X1E>,0                ;7A
01  02D1  6718          .BYTE   ^X01
    02D2  6719          VBUSDATA        <^X1E>,0                ;7B
01  02D3  6720          .BYTE   ^X01
    02D4  6721          VBUSDATA        <^X1E>,1                ;7C
01  02D5  6722          .BYTE   ^X01
    02D6  6723          VBUSDATA        <^X1E>,0                ;7D
01  02D7  6724          .BYTE   ^X01
    02D8  6725          VBUSDATA        <^X1E>,0                ;7E
01  02D9  6726          .BYTE   ^X01
    02DA  6727          VBUSDATA        <^X1E>,1                ;7F
01  02DB  6728          .BYTE   ^X01
    02DC  6729          VBUSDATA        <^X1E>,0                ;80
01  02DD  6730          .BYTE   ^X01
    02DE  6731          VBUSDATA        <^X1E>,0                ;81
01  02DF  6732          .BYTE   ^X01
    02E0  6733          VBUSDATA        <^X1E>,0                ;82
01  02E1  6734          .BYTE   ^X01
    02E2  6735          VBUSDATA        <^X1E>,0                ;83
01  02E3  6736          .BYTE   ^X01
    02E4  6737          VBUSDATA        <^X1E>,0                ;84
01  02E5  6738          .BYTE   ^X01
    02E6  6739          VBUSDATA        <^X1E>,0                ;85
01  02E7  6740          .BYTE   ^X01
    02E8  6741          VBUSDATA        <^X1E>,1                ;86
01  02E9  6742          .BYTE   ^X01
```

```
                    02EA  6743          VBUSDATA        <^X1E>,1                ;87
01                  02EB  6744          .BYTE   ^X01
                    02EC  6745          VBUSDATA        <^X1E>,0                ;88
01                  02ED  6746          .BYTE   ^X01
                    02EE  6747          VBUSDATA        <^X1E>,0                ;89
01                  02EF  6748          .BYTE   ^X01
                    02F0  6749          VBUSDATA        <^X1E>,0                ;8A
01                  02F1  6750          .BYTE   ^X01
                    02F2  6751          VBUSDATA        <^X1E>,0                ;8B
01                  02F3  6752          .BYTE   ^X01
                    02F4  6753          VBUSDATA        <^X1E>,0                ;8C
01                  02F5  6754          .BYTE   ^X01
                    02F6  6755          VBUSDATA        <^X1E>,0                ;8D
01                  02F7  6756          .BYTE   ^X01
                    02F8  6757          VBUSDATA        <^X1E>,0                ;8E
01                  02F9  6758          .BYTE   ^X01
                    02FA  6759          VBUSDATA        <^X1E>,0                ;8F
01                  02FB  6760          .BYTE   ^X01
                    02FC  6761          VBUSDATA        <^X1E>,0                ;90
01                  02FD  6762          .BYTE   ^X01
                    02FE  6763          VBUSDATA        <^X1E>,0                ;91
01                  02FF  6764          .BYTE   ^X01
                    0300  6765          VBUSDATA        <^X1E>,0                ;92
01                  0301  6766          .BYTE   ^X01
                    0302  6767          VBUSDATA        <^X1E>,0                ;93
01                  0303  6768          .BYTE   ^X01
                    0304  6769          VBUSDATA        <^X1E>,1                ;94
01                  0305  6770          .BYTE   ^X01
                    0306  6771          VBUSDATA        <^X1E>,1                ;95
01                  0307  6772          .BYTE   ^X01
                    0308  6773          VBUSDATA        <^X1E>,1                ;96
01                  0309  6774          .BYTE   ^X01
                    030A  6775          VBUSDATA        <^X1E>,1                ;97
01                  030B  6776          .BYTE   ^X01
                    030C  6777          VBUSDATA        <^X1E>,1                ;98
01                  030D  6778          .BYTE   ^X01
                    030E  6779          VBUSDATA        <^X1E>,1                ;99
01                  030F  6780          .BYTE   ^X01
                    0310  6781          VBUSDATA        <^X1E>,1                ;9A
01                  0311  6782          .BYTE   ^X01
                    0312  6783          VBUSDATA        <^X1E>,1                ;9B
01                  0313  6784          .BYTE   ^X01
                    0314  6785          VBUSDATA        <^X1E>,0                ;9C
01                  0315  6786          .BYTE   ^X01
                    0316  6787          VBUSDATA        <^X1E>,0                ;9D
01                  0317  6788          .BYTE   ^X01
                    0318  6789          VBUSDATA        <^X1E>,0                ;9E
01                  0319  6790          .BYTE   ^X01
                    031A  6791          VBUSDATA        <^X1E>,1                ;9F
01                  031B  6792          .BYTE   ^X01
                    031C  6793          VBUSDATA        <^X1E>,0                ;A0
01                  031D  6794          .BYTE   ^X01
                    031E  6795          VBUSDATA        <^X1E>,0                ;A1
01                  031F  6796          .BYTE   ^X01
                    0320  6797          VBUSDATA        <^X1E>,0                ;A2
```

```
                01  0321  6798          .BYTE   ^X01
                    0322  6799          VBUSDATA        <^X1E>,0                ;A3
                01  0323  6800          .BYTE   ^X01
                    0324  6801          VBUSDATA        <^X1E>,0                ;A4
                01  0325  6802          .BYTE   ^X01
                    0326  6803          VBUSDATA        <^X1E>,0                ;A5
                01  0327  6804          .BYTE   ^X01
                    0328  6805          VBUSDATA        <^X1E>,1                ;A6
                01  0329  6806          .BYTE   ^X01
                    032A  6807          VBUSDATA        <^X1E>,1                ;A7
                01  032B  6808          .BYTE   ^X01
                    032C  6809          VBUSDATA        <^X1E>,0                ;A8
                01  032D  6810          .BYTE   ^X01
                    032E  6811          VBUSDATA        <^X1E>,0                ;A9
                01  032F  6812          .BYTE   ^X01
                    0330  6813          VBUSDATA        <^X1E>,0                ;AA
                01  0331  6814          .BYTE   ^X01
                    0332  6815          VBUSDATA        <^X1E>,0                ;AB
                01  0333  6816          .BYTE   ^X01
                    0334  6817          VBUSDATA        <^X1E>,0                ;AC
                01  0335  6818          .BYTE   ^X01
                    0336  6819          VBUSDATA        <^X1E>,0                ;AD
                01  0337  6820          .BYTE   ^X01
                    0338  6821          VBUSDATA        <^X1E>,0                ;AE
                01  0339  6822          .BYTE   ^X01
                    033A  6823          VBUSDATA        <^X1E>,0                ;AF
                01  033B  6824          .BYTE   ^X01
                    033C  6825          VBUSDATA        <^X1E>,0                ;B0
                01  033D  6826          .BYTE   ^X01
                    033E  6827          VBUSDATA        <^X1E>,0                ;B1
                01  033F  6828          .BYTE   ^X01
                    0340  6829          VBUSDATA        <^X1E>,0                ;B2
                01  0341  6830          .BYTE   ^X01
                    0342  6831          VBUSDATA        <^X1E>,0                ;B3
                01  0343  6832          .BYTE   ^X01
                    0344  6833          VBUSDATA        <^X1E>,1                ;B4
                01  0345  6834          .BYTE   ^X01
                    0346  6835          VBUSDATA        <^X1E>,1                ;B5
                01  0347  6836          .BYTE   ^X01
                    0348  6837          VBUSDATA        <^X1E>,1                ;B6
                01  0349  6838          .BYTE   ^X01
                    034A  6839          VBUSDATA        <^X1E>,1                ;B7
                01  034B  6840          .BYTE   ^X01
                    034C  6841          VBUSDATA        <^X1E>,1                ;B8
                01  034D  6842          .BYTE   ^X01
                    034E  6843          VBUSDATA        <^X1E>,1                ;B9
                01  034F  6844          .BYTE   ^X01
                    0350  6845          VBUSDATA        <^X1E>,1                ;BA
                01  0351  6846          .BYTE   ^X01
                    0352  6847          VBUSDATA        <^X1E>,1                ;BB
                01  0353  6848          .BYTE   ^X01
                    0354  6849          VBUSDATA        <^X1E>,1                ;BC
                01  0355  685U          .BYTE   ^X01
                    0356  6851          VBUSDATA        <^X1E>,1                ;BD
                01  0357  6852          .BYTE   ^X01
```

```
                    0358  6853          VBUSDATA        <^X1E>,1                ;BE
01                  0359  6854          .BYTE   ^X01
                    035A  6855          VBUSDATA        <^X1E>,1                ;BF
01                  035B  6856          .BYTE   ^X01
                    035C  6857          VBUSDATA        <^X1E>,0                ;C0
01                  035D  6858          .BYTE   ^X01
                    035E  6859          VBUSDATA        <^X1E>,0                ;C1
01                  035F  6860          .BYTE   ^X01
                    0360  6861          VBUSDATA        <^X1E>,0                ;C2
01                  0361  6862          .BYTE   ^X01
                    0362  6863          VBUSDATA        <^X1E>,0                ;C3
01                  0363  6864          .BYTE   ^X01
                    0364  6865          VBUSDATA        <^X1E>,0                ;C4
01                  0365  6866          .BYTE   ^X01
                    0366  6867          VBUSDATA        <^X1E>,0                ;C5
01                  0367  6868          .BYTE   ^X01
                    0368  6869          VBUSDATA        <^X1E>,1                ;C6
01                  0369  6870          .BYTE   ^X01
                    036A  6871          VBUSDATA        <^X1E>,1                ;C7
01                  036B  6872          .BYTE   ^X01
                    036C  6873          VBUSDATA        <^X1E>,0                ;C8
01                  036D  6874          .BYTE   ^X01
                    036E  6875          VBUSDATA        <^X1E>,0                ;C9
01                  036F  6876          .BYTE   ^X01
                    0370  6877          VBUSDATA        <^X1E>,0                ;CA
01                  0371  6878          .BYTE   ^X01
                    0372  6879          VBUSDATA        <^X1E>,0                ;CB
01                  0373  6880          .BYTE   ^X01
                    0374  6881          VBUSDATA        <^X1E>,0                ;CC
01                  0375  6882          .BYTE   ^X01
                    0376  6883          VBUSDATA        <^X1E>,0                ;CD
01                  0377  6884          .BYTE   ^X01
                    0378  6885          VBUSDATA        <^X1E>,0                ;CE
01                  0379  6886          .BYTE   ^X01
                    037A  6887          VBUSDATA        <^X1E>,0                ;CF
01                  037B  6888          .BYTE   ^X01
                    037C  6889          VBUSDATA        <^X1E>,0                ;D0
01                  037D  6890          .BYTE   ^X01
                    037E  6891          VBUSDATA        <^X1E>,0                ;D1
01                  037F  6892          .BYTE   ^X01
                    0380  6893          VBUSDATA        <^X1E>,0                ;D2
01                  0381  6894          .BYTE   ^X01
                    0382  6895          VBUSDATA        <^X1E>,0                ;D3
01                  0383  6896          .BYTE   ^X01
                    0384  6897          VBUSDATA        <^X1E>,1                ;D4
01                  0385  6898          .BYTE   ^X01
                    0386  6899          VBUSDATA        <^X1E>,1                ;D5
01                  0387  6900          .BYTE   ^X01
                    0388  6901          VBUSDATA        <^X1E>,1                ;D6
01                  0389  6902          .BYTE   ^X01
                    038A  6903          VBUSDATA        <^X1E>,1                ;D7
01                  038B  6904          .BYTE   ^X01
                    038C  6905          VBUSDATA        <^X1E>,0                ;D8
01                  038D  6906          .BYTE   ^X01
                    038E  6907          VBUSDATA        <^X1E>,0                ;D9
```

```
01  038F  6908          .BYTE   ^X01
    0390  6909          VBUSDATA        <^X1E>,0                ;DA
01  0391  6910          .BYTE   ^X01
    0392  6911          VBUSDATA        <^X1E>,1                ;DB
01  0393  6912          .BYTE   ^X01
    0394  6913          VBUSDATA        <^X1E>,1                ;DC
01  0395  6914          .BYTE   ^X01
    0396  6915          VBUSDATA        <^X1E>,1                ;DD
01  0397  6916          .BYTE   ^X01
    0398  6917          VBUSDATA        <^X1E>,0                ;DE
01  0399  6918          .BYTE   ^X01
    039A  6919          VBUSDATA        <^X1E>,1                ;DF
01  039B  6920          .BYTE   ^X01
    039C  6921          VBUSDATA        <^X1E>,0                ;E0
01  039D  6922          .BYTE   ^X01
    039E  6923          VBUSDATA        <^X1E>,0                ;E1
01  039F  6924          .BYTE   ^X01
    03A0  6925          VBUSDATA        <^X1E>,0                ;E2
01  03A1  6926          .BYTE   ^X01
    03A2  6927          VBUSDATA        <^X1E>,0                ;E3
01  03A3  6928          .BYTE   ^X01
    03A4  6929          VBUSDATA        <^X1E>,0                ;E4
01  03A5  6930          .BYTE   ^X01
    03A6  6931          VBUSDATA        <^X1E>,0                ;E5
01  03A7  6932          .BYTE   ^X01
    03A8  6933          VBUSDATA        <^X1E>,0                ;E6
01  03A9  6934          .BYTE   ^X01
    03AA  6935          VBUSDATA        <^X1E>,0                ;E7
01  03AB  6936          .BYTE   ^X01
    03AC  6937          VBUSDATA        <^X1E>,1                ;E8
01  03AD  6938          .BYTE   ^X01
    03AE  6939          VBUSDATA        <^X1E>,1                ;E9
01  03AF  6940          .BYTE   ^X01
    03B0  6941          VBUSDATA        <^X1E>,0                ;EA
01  03B1  6942          .BYTE   ^X01
    03B2  6943          VBUSDATA        <^X1E>,0                ;EB
01  03B3  6944          .BYTE   ^X01
    03B4  6945          VBUSDATA        <^X1E>,0                ;EC
01  03B5  6946          .BYTE   ^X01
    03B6  6947          VBUSDATA        <^X1E>,0                ;ED
01  03B7  6948          .BYTE   ^X01
    03B8  6949          VBUSDATA        <^X1E>,0                ;EE
01  03B9  6950          .BYTE   ^X01
    03BA  6951          VBUSDATA        <^X1E>,0                ;EF
01  03BB  6952          .BYTE   ^X01
    03BC  6953          VBUSDATA        <^X1E>,0                ;F0
01  03BD  6954          .BYTE   ^X01
    03BE  6955          VBUSDATA        <^X1E>,0                ;F1
01  03BF  6956          .BYTE   ^X01
    03C0  6957          VBUSDATA        <^X1E>,0                ;F2
01  03C1  6958          .BYTE   ^X01
    03C2  6959          VBUSDATA        <^X1E>,0                ;F3
01  03C3  6960          .BYTE   ^X01
    03C4  6961          VBUSDATA        <^X1E>,1                ;F4
01  03C5  6962          .BYTE   ^X01
```

```
                    03C6  6963          VBUSDATA        <^X1E>,1                ;F5
 01                 03C7  6964          .BYTE   ^X01
                    03C8  6965          VBUSDATA        <^X1E>,0                ;F6
 01                 03C9  6966          .BYTE   ^X01
                    03CA  6967          VBUSDATA        <^X1E>,0                ;F7
 01                 03CB  6968          .BYTE   ^X01
                    03CC  6969          VBUSDATA        <^X1E>,1                ;F8
 01                 03CD  6970          .BYTE   ^X01
                    03CE  6971          VBUSDATA        <^X1E>,0                ;F9
 01                 03CF  6972          .BYTE   ^X01
                    03D0  6973          VBUSDATA        <^X1E>,0                ;FA
 01                 03D1  6974          .BYTE   ^X01
                    03D2  6975          VBUSDATA        <^X1E>,0                ;FB
 01                 03D3  6976          .BYTE   ^X01
                    03D4  6977          VBUSDATA        <^X1E>,1                ;FC
 01                 03D5  6978          .BYTE   ^X01
                    03D6  6979          VBUSDATA        <^X1E>,1                ;FD
 01                 03D7  6980          .BYTE   ^X01
                    03D8  6981          VBUSDATA        <^X1E>,1                ;FE
 01                 03D9  6982          .BYTE   ^X01
                    03DA  6983          VBUSDATA        <^X1E>,1                ;FF
                    03DB  6984
                    03DB  6985
                    03DB  6986 ;
                    03DB  6987 ; Expected VBUS values for Subtest 3
                    03DB  6988 ;
                    03DB  6989
 01                 03DB  6990 31$:     .BYTE   ^X01
                    03DC  6991          VBUSDATA        <^X1E>,1                ;00
 01                 03DD  6992          .BYTE   ^X01
                    03DE  6993          VBUSDATA        <^X1E>,1                ;01
 01                 03DF  6994          .BYTE   ^X01
                    03E0  6995          VBUSDATA        <^X1E>,1                ;02
 01                 03E1  6996          .BYTE   ^X01
                    03E2  6997          VBUSDATA        <^X1E>,1                ;03
 01                 03E3  6998          .BYTE   ^X01
                    03E4  6999          VBUSDATA        <^X1E>,1                ;04
 01                 03E5  7000          .BYTE   ^X01
                    03E6  7001          VBUSDATA        <^X1E>,1                ;05
 01                 03E7  7002          .BYTE   ^X01
                    03E8  7003          VBUSDATA        <^X1E>,1                ;06
 01                 03E9  7004          .BYTE   ^X01
                    03EA  7005          VBUSDATA        <^X1E>,1                ;07
 01                 03EB  7006          .BYTE   ^X01
                    03EC  7007          VBUSDATA        <^X1E>,1                ;08
 01                 03ED  7008          .BYTE   ^X01
                    03EE  7009          VBUSDATA        <^X1E>,1                ;09
 01                 03EF  7010          .BYTE   ^X01
                    03F0  7011          VBUSDATA        <^X1E>,0                ;0A
 01                 03F1  7012          .BYTE   ^X01
                    03F2  7013          VBUSDATA        <^X1E>,1                ;0B
 01                 03F3  7014          .BYTE   ^X01
                    03F4  7015          VBUSDATA        <^X1E>,0                ;0C
 01                 03F5  7016          .BYTE   ^X01
                    03F6  7017          VBUSDATA        <^X1E>,0                ;0D
```

```
01  03F7  7018          .BYTE   ^X01
    03F8  7019          VBUSDATA        <^X1E>,0                ;0E
01  03F9  7020          .BYTE   ^X01
    03FA  7021          VBUSDATA        <^X1E>,0                ;0F
01  03FB  7022          .BYTE   ^X01
    03FC  7023          VBUSDATA        <^X1E>,1                ;10
01  03FD  7024          .BYTE   ^X01
    03FE  7025          VBUSDATA        <^X1E>,1                ;11
01  03FF  7026          .BYTE   ^X01
    0400  7027          VBUSDATA        <^X1E>,1                ;12
01  0401  7028          .BYTE   ^X01
    0402  7029          VBUSDATA        <^X1E>,1                ;13
01  0403  7030          .BYTE   ^X01
    0404  7031          VBUSDATA        <^X1E>,1                ;14
01  0405  7032          .BYTE   ^X01
    0406  7033          VBUSDATA        <^X1E>,1                ;15
01  0407  7034          .BYTE   ^X01
    0408  7035          VBUSDATA        <^X1E>,1                ;16
01  0409  7036          .BYTE   ^X01
    040A  7037          VBUSDATA        <^X1E>,1                ;17
01  040B  7038          .BYTE   ^X01
    040C  7039          VBUSDATA        <^X1E>,1                ;18
01  040D  7040          .BYTE   ^X01
    040E  7041          VBUSDATA        <^X1E>,1                ;19
01  040F  7042          .BYTE   ^X01
    0410  7043          VBUSDATA        <^X1E>,1                ;1A
01  0411  7044          .BYTE   ^X01
    0412  7045          VBUSDATA        <^X1E>,1                ;1B
01  0413  7046          .BYTE   ^X01
    0414  7047          VBUSDATA        <^X1E>,1                ;1C
01  0415  7048          .BYTE   ^X01
    0416  7049          VBUSDATA        <^X1E>,1                ;1D
01  0417  7050          .BYTE   ^X01
    0418  7051          VBUSDATA        <^X1E>,1                ;1E
01  0419  7052          .BYTE   ^X01
    041A  7053          VBUSDATA        <^X1E>,1                ;1F
01  041B  7054          .BYTE   ^X01
    041C  7055          VBUSDATA        <^X1E>,1                ;20
01  041D  7056          .BYTE   ^X01
    041E  7057          VBUSDATA        <^X1E>,1                ;21
01  041F  7058          .BYTE   ^X01
    0420  7059          VBUSDATA        <^X1E>,1                ;22
01  0421  7060          .BYTE   ^X01
    0422  7061          VBUSDATA        <^X1E>,1                ;23
01  0423  7062          .BYTE   ^X01
    0424  7063          VBUSDATA        <^X1E>,1                ;24
01  0425  7064          .BYTE   ^X01
    0426  7065          VBUSDATA        <^X1E>,1                ;25
01  0427  7066          .BYTE   ^X01
    0428  7067          VBUSDATA        <^X1E>,1                ;26
01  0429  7068          .BYTE   ^X01
    042A  7069          VBUSDATA        <^X1E>,1                ;27
01  042B  7070          .BYTE   ^X01
    042C  7071          VBUSDATA        <^X1E>,1                ;28
01  042D  7072          .BYTE   ^X01
```

```
                042E  7073          VBUSDATA        <^X1E>,1                ;29
            01  042F  7074          .BYTE   ^X01
                0430  7075          VBUSDATA        <^X1E>,1                ;2A
            01  0431  7076          .BYTE   ^X01
                0432  7077          VBUSDATA        <^X1E>,1                ;2B
            01  0433  7078          .BYTE   ^X01
                0434  7079          VBUSDATA        <^X1E>,1                ;2C
            01  0435  7080          .BYTE   ^X01
                0436  7081          VBUSDATA        <^X1E>,1                ;2D
            01  0437  7082          .BYTE   ^X01
                0438  7083          VBUSDATA        <^X1E>,1                ;2E
            01  0439  7084          .BYTE   ^X01
                043A  7085          VBUSDATA        <^X1E>,1                ;2F
            01  043B  7086          .BYTE   ^X01
                043C  7087          VBUSDATA        <^X1E>,1                ;30
            01  043D  7088          .BYTE   ^X01
                043E  7089          VBUSDATA        <^X1E>,1                ;31
            01  043F  7090          .BYTE   ^X01
                0440  7091          VBUSDATA        <^X1E>,1                ;32
            01  0441  7092          .BYTE   ^X01
                0442  7093          VBUSDATA        <^X1E>,0                ;33
            01  0443  7094          .BYTE   ^X01
                0444  7095          VBUSDATA        <^X1E>,1                ;34
            01  0445  7096          .BYTE   ^X01
                0446  7097          VBUSDATA        <^X1E>,1                ;35
            01  0447  7098          .BYTE   ^X01
                0448  7099          VBUSDATA        <^X1E>,1                ;36
            01  0449  7100          .BYTE   ^X01
                044A  7101          VBUSDATA        <^X1E>,1                ;37
            01  044B  7102          .BYTE   ^X01
                044C  7103          VBUSDATA        <^X1E>,1                ;38
            01  044D  7104          .BYTE   ^X01
                044E  7105          VBUSDATA        <^X1E>,1                ;39
            01  044F  7106          .BYTE   ^X01
                0450  7107          VBUSDATA        <^X1E>,1                ;3A
            01  0451  7108          .BYTE   ^X01
                0452  7109          VBUSDATA        <^X1E>,1                ;3B
            01  0453  7110          .BYTE   ^X01
                0454  7111          VBUSDATA        <^X1E>,1                ;3C
            01  0455  7112          .BYTE   ^X01
                0456  7113          VBUSDATA        <^X1E>,0                ;3D
            01  0457  7114          .BYTE   ^X01
                0458  7115          VBUSDATA        <^X1E>,0                ;3E
            01  0459  7116          .BYTE   ^X01
                045A  7117          VBUSDATA        <^X1E>,1                ;3F
            01  045B  7118          .BYTE   ^X01
                045C  7119          VBUSDATA        <^X1E>,0                ;40
            01  045D  7120          .BYTE   ^X01
                045E  7121          VBUSDATA        <^X1E>,0                ;41
            01  045F  7122          .BYTE   ^X01
                0460  7123          VBUSDATA        <^X1E>,0                ;42
            01  0461  7124          .BYTE   ^X01
                0462  7125          VBUSDATA        <^X1E>,0                ;43
            01  0463  7126          .BYTE   ^X01
                0464  7127          VBUSDATA        <^X1E>,0                ;44
```

```
01  0465  7128          .BYTE   ^X01
    0466  7129          VBUSDATA        <^X1E>,0                ;45
01  0467  7130          .BYTE   ^X01
    0468  7131          VBUSDATA        <^X1E>,0                ;46
01  0469  7132          .BYTE   ^X01
    046A  7133          VBUSDATA        <^X1E>,0                ;47
01  046B  7134          .BYTE   ^X01
    046C  7135          VBUSDATA        <^X1E>,1                ;48
01  046D  7136          .BYTE   ^X01
    046E  7137          VBUSDATA        <^X1E>,1                ;49
01  046F  7138          .BYTE   ^X01
    0470  7139          VBUSDATA        <^X1E>,1                ;4A
01  0471  7140          .BYTE   ^X01
    0472  7141          VBUSDATA        <^X1E>,1                ;4B
01  0473  7142          .BYTE   ^X01
    0474  7143          VBUSDATA        <^X1E>,1                ;4C
01  0475  7144          .BYTE   ^X01
    0476  7145          VBUSDATA        <^X1E>,1                ;4D
01  0477  7146          .BYTE   ^X01
    0478  7147          VBUSDATA        <^X1E>,1                ;4E
01  0479  7148          .BYTE   ^X01
    047A  7149          VBUSDATA        <^X1E>,0                ;4F
01  047B  7150          .BYTE   ^X01
    047C  7151          VBUSDATA        <^X1E>,0                ;50
01  047D  7152          .BYTE   ^X01
    047E  7153          VBUSDATA        <^X1E>,0                ;51
01  047F  7154          .BYTE   ^X01
    0480  7155          VBUSDATA        <^X1E>,0                ;52
01  0481  7156          .BYTE   ^X01
    0482  7157          VBUSDATA        <^X1E>,1                ;53
01  0483  7158          .BYTE   ^X01
    0484  7159          VBUSDATA        <^X1E>,0                ;54
01  0485  7160          .BYTE   ^X01
    0486  7161          VBUSDATA        <^X1E>,0                ;55
01  0487  7162          .BYTE   ^X01
    0488  7163          VBUSDATA        <^X1E>,1                ;56
01  0489  7164          .BYTE   ^X01
    048A  7165          VBUSDATA        <^X1E>,1                ;57
01  048B  7166          .BYTE   ^X01
    048C  7167          VBUSDATA        <^X1E>,0                ;58
01  048D  7168          .BYTE   ^X01
    048E  7169          VBUSDATA        <^X1E>,1                ;59
01  048F  7170          .BYTE   ^X01
    0490  7171          VBUSDATA        <^X1E>,1                ;5A
01  0491  7172          .BYTE   ^X01
    0492  7173          VBUSDATA        <^X1E>,1                ;5B
01  0493  7174          .BYTE   ^X01
    0494  7175          VBUSDATA        <^X1E>,0                ;5C
01  0495  7176          .BYTE   ^X01
    0496  7177          VBUSDATA        <^X1E>,0                ;5D
01  0497  7178          .BYTE   ^X01
    0498  7179          VBUSDATA        <^X1E>,0                ;5E
01  0499  7180          .BYTE   ^X01
    049A  7181          VBUSDATA        <^X1E>,0                ;5F
01  049B  7182          .BYTE   ^X01
```

```
                049C  7183          VBUSDATA        <^X1E>,0                ;60
            01  049D  7184          .BYTE   ^X01
                049E  7185          VBUSDATA        <^X1E>,0                ;61
            01  049F  7186          .BYTE   ^X01
                04A0  7187          VBUSDATA        <^X1E>,0                ;62
            01  04A1  7188          .BYTE   ^X01
                04A2  7189          VBUSDATA        <^X1E>,0                ;63
            01  04A3  7190          .BYTE   ^X01
                04A4  7191          VBUSDATA        <^X1E>,0                ;64
            01  04A5  7192          .BYTE   ^X01
                04A6  7193          VBUSDATA        <^X1E>,0                ;65
            01  04A7  7194          .BYTE   ^X01
                04A8  7195          VBUSDATA        <^X1E>,0                ;66
            01  04A9  7196          .BYTE   ^X01
                04AA  7197          VBUSDATA        <^X1E>,0                ;67
            01  04AB  7198          .BYTE   ^X01
                04AC  7199          VBUSDATA        <^X1E>,1                ;68
            01  04AD  7200          .BYTE   ^X01
                04AE  7201          VBUSDATA        <^X1E>,1                ;69
            01  04AF  7202          .BYTE   ^X01
                04B0  7203          VBUSDATA        <^X1E>,1                ;6A
            01  04B1  7204          .BYTE   ^X01
                04B2  7205          VBUSDATA        <^X1E>,1                ;6B
            01  04B3  7206          .BYTE   ^X01
                04B4  7207          VBUSDATA        <^X1E>,1                ;6C
            01  04B5  7208          .BYTE   ^X01
                04B6  7209          VBUSDATA        <^X1E>,1                ;6D
            01  04B7  7210          .BYTE   ^X01
                04B8  7211          VBUSDATA        <^X1E>,1                ;6E
            01  04B9  7212          .BYTE   ^X01
                04BA  7213          VBUSDATA        <^X1E>,1                ;6F
            01  04BB  7214          .BYTE   ^X01
                04BC  7215          VBUSDATA        <^X1E>,0                ;70
            01  04BD  7216          .BYTE   ^X01
                04BE  7217          VBUSDATA        <^X1E>,0                ;71
            01  04BF  7218          .BYTE   ^X01
                04C0  7219          VBUSDATA        <^X1E>,0                ;72
            01  04C1  7220          .BYTE   ^X01
                04C2  7221          VBUSDATA        <^X1E>,1                ;73
            01  04C3  7222          .BYTE   ^X01
                04C4  7223          VBUSDATA        <^X1E>,0                ;74
            01  04C5  7224          .BYTE   ^X01
                04C6  7225          VBUSDATA        <^X1E>,1                ;75
            01  04C7  7226          .BYTE   ^X01
                04C8  7227          VBUSDATA        <^X1E>,1                ;76
            01  04C9  7228          .BYTE   ^X01
                04CA  7229          VBUSDATA        <^X1E>,1                ;77
            01  04CB  7230          .BYTE   ^X01
                04CC  7231          VBUSDATA        <^X1E>,0                ;78
            01  04CD  7232          .BYTE   ^X01
                04CE  7233          VBUSDATA        <^X1E>,0                ;79
            01  04CF  7234          .BYTE   ^X01
                04D0  7235          VBUSDATA        <^X1E>,0                ;7A
            01  04D1  7236          .BYTE   ^X01
                04D2  7237          VBUSDATA        <^X1E>,0                ;7B
```

```
01  04D3  7238          .BYTE   ^X01
    04D4  7239          VBUSDATA        <^X1E>,1                ;7C
01  04D5  7240          .BYTE   ^X01
    04D6  7241          VBUSDATA        <^X1E>,0                ;7D
01  04D7  7242          .BYTE   ^X01
    04D8  7243          VBUSDATA        <^X1E>,0                ;7E
01  04D9  7244          .BYTE   ^X01
    04DA  7245          VBUSDATA        <^X1E>,1                ;7F
01  04DB  7246          .BYTE   ^X01
    04DC  7247          VBUSDATA        <^X1E>,0                ;80
01  04DD  7248          .BYTE   ^X01
    04DE  7249          VBUSDATA        <^X1E>,0                ;81
01  04DF  7250          .BYTE   ^X01
    04E0  7251          VBUSDATA        <^X1E>,0                ;82
01  04E1  7252          .BYTE   ^X01
    04E2  7253          VBUSDATA        <^X1E>,0                ;83
01  04E3  7254          .BYTE   ^X01
    04E4  7255          VBUSDATA        <^X1E>,0                ;84
01  04E5  7256          .BYTE   ^X01
    04E6  7257          VBUSDATA        <^X1E>,0                ;85
01  04E7  7258          .BYTE   ^X01
    04E8  7259          VBUSDATA        <^X1E>,1                ;86
01  04E9  7260          .BYTE   ^X01
    04EA  7261          VBUSDATA        <^X1E>,1                ;87
01  04EB  7262          .BYTE   ^X01
    04EC  7263          VBUSDATA        <^X1E>,0                ;88
01  04ED  7264          .BYTE   ^X01
    04EE  7265          VBUSDATA        <^X1E>,0                ;89
01  04EF  7266          .BYTE   ^X01
    04F0  7267          VBUSDATA        <^X1E>,0                ;8A
01  04F1  7268          .BYTE   ^X01
    04F2  7269          VBUSDATA        <^X1E>,0                ;8B
01  04F3  7270          .BYTE   ^X01
    04F4  7271          VBUSDATA        <^X1E>,0                ;8C
01  04F5  7272          .BYTE   ^X01
    04F6  7273          VBUSDATA        <^X1E>,0                ;8D
01  04F7  7274          .BYTE   ^X01
    04F8  7275          VBUSDATA        <^X1E>,0                ;8E
01  04F9  7276          .BYTE   ^X01
    04FA  7277          VBUSDATA        <^X1E>,0                ;8F
01  04FB  7278          .BYTE   ^X01
    04FC  7279          VBUSDATA        <^X1E>,0                ;90
01  04FD  7280          .BYTE   ^X01
    04FE  7281          VBUSDATA        <^X1E>,0                ;91
01  04FF  7282          .BYTE   ^X01
    0500  7283          VBUSDATA        <^X1E>,0                ;92
01  0501  7284          .BYTE   ^X01
    0502  7285          VBUSDATA        <^X1E>,0                ;93
01  0503  7286          .BYTE   ^X01
    0504  7287          VBUSDATA        <^X1E>,1                ;94
01  0505  7288          .BYTE   ^X01
    0506  7289          VBUSDATA        <^X1E>,1                ;95
01  0507  7290          .BYTE   ^X01
    0508  7291          VBUSDATA        <^X1E>,1                ;96
01  0509  7292          .BYTE   ^X01
```

```
                050A  7293          VBUSDATA        <^X1E>,1                ;97
            01  050B  7294          .BYTE   ^X01
                050C  7295          VBUSDATA        <^X1E>,1                ;98
            01  050D  7296          .BYTE   ^X01
                050E  7297          VBUSDATA        <^X1E>,1                ;99
            01  050F  7298          .BYTE   ^X01
                0510  7299          VBUSDATA        <^X1E>,1                ;9A
            01  0511  7300          .BYTE   ^X01
                0512  7301          VBUSDATA        <^X1E>,1                ;9B
            01  0513  7302          .BYTE   ^X01
                0514  7303          VBUSDATA        <^X1E>,0                ;9C
            01  0515  7304          .BYTE   ^X01
                0516  7305          VBUSDATA        <^X1E>,0                ;9D
            01  0517  7306          .BYTE   ^X01
                0518  7307          VBUSDATA        <^X1E>,0                ;9E
            01  0519  7308          .BYTE   ^X01
                051A  7309          VBUSDATA        <^X1E>,1                ;9F
            01  051B  7310          .BYTE   ^X01
                051C  7311          VBUSDATA        <^X1E>,0                ;A0
            01  051D  7312          .BYTE   ^X01
                051E  7313          VBUSDATA        <^X1E>,0                ;A1
            01  051F  7314          .BYTE   ^X01
                0520  7315          VBUSDATA        <^X1E>,0                ;A2
            01  0521  7316          .BYTE   ^X01
                0522  7317          VBUSDATA        <^X1E>,0                ;A3
            01  0523  7318          .BYTE   ^X01
                0524  7319          VBUSDATA        <^X1E>,0                ;A4
            01  0525  7320          .BYTE   ^X01
                0526  7321          VBUSDATA        <^X1E>,0                ;A5
            01  0527  7322          .BYTE   ^X01
                0528  7323          VBUSDATA        <^X1E>,1                ;A6
            01  0529  7324          .BYTE   ^X01
                052A  7325          VBUSDATA        <^X1E>,1                ;A7
            01  052B  7326          .BYTE   ^X01
                052C  7327          VBUSDATA        <^X1E>,0                ;A8
            01  052D  7328          .BYTE   ^X01
                052E  7329          VBUSDATA        <^X1E>,0                ;A9
            01  052F  7330          .BYTE   ^X01
                0530  7331          VBUSDATA        <^X1E>,0                ;AA
            01  0531  7332          .BYTE   ^X01
                0532  7333          VBUSDATA        <^X1E>,0                ;AB
            01  0533  7334          .BYTE   ^X01
                0534  7335          VBUSDATA        <^X1E>,0                ;AC
            01  0535  7336          .BYTE   ^X01
                0536  7337          VBUSDATA        <^X1E>,0                ;AD
            01  0537  7338          .BYTE   ^X01
                0538  7339          VBUSDATA        <^X1E>,0                ;AE
            01  0539  7340          .BYTE   ^X01
                053A  7341          VBUSDATA        <^X1E>,0                ;AF
            01  053B  7342          .BYTE   ^X01
                053C  7343          VBUSDATA        <^X1E>,0                ;B0
            01  053D  7344          .BYTE   ^X01
                053E  7345          VBUSDATA        <^X1E>,0                ;B1
            01  053F  7346          .BYTE   ^X01
                0540  7347          VBUSDATA        <^X1E>,0                ;B2
```

```
01  0541  7348          .BYTE   ^X01
    0542  7349          VBUSDATA        <^X1E>,0                ;B3
01  0543  7350          .BYTE   ^X01
    0544  7351          VBUSDATA        <^X1E>,1                ;B4
01  0545  7352          .BYTE   ^X01
    0546  7353          VBUSDATA        <^X1E>,1                ;B5
01  0547  7354          .BYTE   ^X01
    0548  7355          VBUSDATA        <^X1E>,1                ;B6
01  0549  7356          .BYTE   ^X01
    054A  7357          VBUSDATA        <^X1E>,1                ;B7
01  054B  7358          .BYTE   ^X01
    054C  7359          VBUSDATA        <^X1E>,1                ;B8
01  054D  7360          .BYTE   ^X01
    054E  7361          VBUSDATA        <^X1E>,1                ;B9
01  054F  7362          .BYTE   ^X01
    0550  7363          VBUSDATA        <^X1E>,1                ;BA
01  0551  7364          .BYTE   ^X01
    0552  7365          VBUSDATA        <^X1E>,1                ;BB
01  0553  7366          .BYTE   ^X01
    0554  7367          VBUSDATA        <^X1E>,1                ;BC
01  0555  7368          .BYTE   ^X01
    0556  7369          VBUSDATA        <^X1E>,1                ;BD
01  0557  7370          .BYTE   ^X01
    0558  7371          VBUSDATA        <^X1E>,1                ;BE
01  0559  7372          .BYTE   ^X01
    055A  7373          VBUSDATA        <^X1E>,1                ;BF
01  055B  7374          .BYTE   ^X01
    055C  7375          VBUSDATA        <^X1E>,0                ;C0
01  055D  7376          .BYTE   ^X01
    055E  7377          VBUSDATA        <^X1E>,0                ;C1
01  055F  7378          .BYTE   ^X01
    0560  7379          VBUSDATA        <^X1E>,0                ;C2
01  0561  7380          .BYTE   ^X01
    0562  7381          VBUSDATA        <^X1E>,0                ;C3
01  0563  7382          .BYTE   ^X01
    0564  7383          VBUSDATA        <^X1E>,0                ;C4
01  0565  7384          .BYTE   ^X01
    0566  7385          VBUSDATA        <^X1E>,0                ;C5
01  0567  7386          .BYTE   ^X01
    0568  7387          VBUSDATA        <^X1E>,1                ;C6
01  0569  7388          .BYTE   ^X01
    056A  7389          VBUSDATA        <^X1E>,1                ;C7
01  056B  7390          .BYTE   ^X01
    056C  7391          VBUSDATA        <^X1E>,0                ;C8
01  056D  7392          .BYTE   ^X01
    056E  7393          VBUSDATA        <^X1E>,0                ;C9
01  056F  7394          .BYTE   ^X01
    0570  7395          VBUSDATA        <^X1E>,0                ;CA
01  0571  7396          .BYTE   ^X01
    0572  7397          VBUSDATA        <^X1E>,0                ;CB
01  0573  7398          .BYTE   ^X01
    0574  7399          VBUSDATA        <^X1E>,0                ;CC
01  0575  7400          .BYTE   ^X01
    0576  7401          VBUSDATA        <^X1E>,0                ;CD
01  0577  7402          .BYTE   ^X01
```

```
                0578  7403          VBUSDATA        <^X1E>,0                ;CE
01              0579  7404          .BYTE   ^X01
                057A  7405          VBUSDATA        <^X1E>,0                ;CF
01              057B  7406          .BYTE   ^X01
                057C  7407          VBUSDATA        <^X1E>,0                ;D0
01              057D  7408          .BYTE   ^X01
                057E  7409          VBUSDATA        <^X1E>,0                ;D1
01              057F  7410          .BYTE   ^X01
                0580  7411          VBUSDATA        <^X1E>,0                ;D2
01              0581  7412          .BYTE   ^X01
                0582  7413          VBUSDATA        <^X1E>,0                ;D3
01              0583  7414          .BYTE   ^X01
                0584  7415          VBUSDATA        <^X1E>,1                ;D4
01              0585  7416          .BYTE   ^X01
                0586  7417          VBUSDATA        <^X1E>,1                ;D5
01              0587  7418          .BYTE   ^X01
                0588  7419          VBUSDATA        <^X1E>,1                ;D6
01              0589  7420          .BYTE   ^X01
                058A  7421          VBUSDATA        <^X1E>,1                ;D7
01              058B  7422          .BYTE   ^X01
                058C  7423          VBUSDATA        <^X1E>,0                ;D8
01              058D  7424          .BYTE   ^X01
                058E  7425          VBUSDATA        <^X1E>,0                ;D9
01              058F  7426          .BYTE   ^X01
                0590  7427          VBUSDATA        <^X1E>,0                ;DA
01              0591  7428          .BYTE   ^X01
                0592  7429          VBUSDATA        <^X1E>,1                ;DB
01              0593  7430          .BYTE   ^X01
                0594  7431          VBUSDATA        <^X1E>,1                ;DC
01              0595  7432          .BYTE   ^X01
                0596  7433          VBUSDATA        <^X1E>,1                ;DD
01              0597  7434          .BYTE   ^X01
                0598  7435          VBUSDATA        <^X1E>,0                ;DE
01              0599  7436          .BYTE   ^X01
                059A  7437          VBUSDATA        <^X1E>,1                ;DF
01              059B  7438          .BYTE   ^X01
                059C  7439          VBUSDATA        <^X1E>,0                ;E0
01              059D  7440          .BYTE   ^X01
                059E  7441          VBUSDATA        <^X1E>,0                ;E1
01              059F  7442          .BYTE   ^X01
                05A0  7443          VBUSDATA        <^X1E>,0                ;E2
01              05A1  7444          .BYTE   ^X01
                05A2  7445          VBUSDATA        <^X1E>,0                ;E3
01              05A3  7446          .BYTE   ^X01
                05A4  7447          VBUSDATA        <^X1E>,0                ;E4
01              05A5  7448          .BYTE   ^X01
                05A6  7449          VBUSDATA        <^X1E>,0                ;E5
01              05A7  7450          .BYTE   ^X01
                05A8  7451          VBUSDATA        <^X1E>,0                ;E6
01              05A9  7452          .BYTE   ^X01
                05AA  7453          VBUSDATA        <^X1E>,0                ;E7
01              05AB  7454          .BYTE   ^X01
                05AC  7455          VBUSDATA        <^X1E>,1                ;E8
01              05AD  7456          .BYTE   ^X01
                05AE  7457          VBUSDATA        <^X1E>,1                ;E9
```

```
                    01  05AF  7458          .BYTE   ^X01
                        05B0  7459          VBUSDATA        <^X1E>,0                ;EA
                    01  05B1  7460          .BYTE   ^X01
                        05B2  7461          VBUSDATA        <^X1E>,0                ;EB
                    01  05B3  7462          .BYTE   ^X01
                        05B4  7463          VBUSDATA        <^X1E>,0                ;EC
                    01  05B5  7464          .BYTE   ^X01
                        05B6  7465          VBUSDATA        <^X1E>,0                ;ED
                    01  05B7  7466          .BYTE   ^X01
                        05B8  7467          VBUSDATA        <^X1E>,0                ;EE
                    01  05B9  7468          .BYTE   ^X01
                        05BA  7469          VBUSDATA        <^X1E>,0                ;EF
                    01  05BB  7470          .BYTE   ^X01
                        05BC  7471          VBUSDATA        <^X1E>,0                ;F0
                    01  05BD  7472          .BYTE   ^X01
                        05BE  7473          VBUSDATA        <^X1E>,0                ;F1
                    01  05BF  7474          .BYTE   ^X01
                        05C0  7475          VBUSDATA        <^X1E>,0                ;F2
                    01  05C1  7476          .BYTE   ^X01
                        05C2  7477          VBUSDATA        <^X1E>,0                ;F3
                    01  05C3  7478          .BYTE   ^X01
                        05C4  7479          VBUSDATA        <^X1E>,1                ;F4
                    01  05C5  7480          .BYTE   ^X01
                        05C6  7481          VBUSDATA        <^X1E>,1                ;F5
                    01  05C7  7482          .BYTE   ^X01
                        05C8  7483          VBUSDATA        <^X1E>,0                ;F6
                    01  05C9  7484          .BYTE   ^X01
                        05CA  7485          VBUSDATA        <^X1E>,0                ;F7
                    01  05CB  7486          .BYTE   ^X01
                        05CC  7487          VBUSDATA        <^X1E>,1                ;F8
                    01  05CD  7488          .BYTE   ^X01
                        05CE  7489          VBUSDATA        <^X1E>,0                ;F9
                    01  05CF  7490          .BYTE   ^X01
                        05D0  7491          VBUSDATA        <^X1E>,0                ;FA
                    01  05D1  7492          .BYTE   ^X01
                        05D2  7493          VBUSDATA        <^X1E>,0                ;FB
                    01  05D3  7494          .BYTE   ^X01
                        05D4  7495          VBUSDATA        <^X1E>,1                ;FC
                    01  05D5  7496          .BYTE   ^X01
                        05D6  7497          VBUSDATA        <^X1E>,1                ;FD
                    01  05D7  7498          .BYTE   ^X01
                        05D8  7499          VBUSDATA        <^X1E>,1                ;FE
                    01  05D9  7500          .BYTE   ^X01
                        05DA  7501          VBUSDATA        <^X1E>,1                ;FF
                        05DB  7502
                        05DB  7503 BEGIN_CPU_DATA
                        0002  7504
                        0002  7505 DCS_DATA
                        0001  7506
                        0001  7507 ; Micro-code to perform the instruction stream initialization and IRD1.
                        0001  7508
U 00, 7700,C800,0364,0300,0470,1801   0001  7509 ;% 00   NOP                                     ; get cpu up to speed
U 01, 5700,C840,0364,0308,0470,1802   000C  7513 ;% 01   R[TEMP2]_RDM                           ; put opcode in R[TEMP2]
U 02, 7700,F800,4ECF,4FFF,8470,1803   0017  7517 ;% 02   LONLIT_[62616000]                     ; load opspecs in lonlit reg
U 03, 7700,8800,5BE5,03D4,0470,1804   0022  7521 ;% 03   Q_R[LONLIT]                           ; move opspecs to Q register
```

```
U 04. 7700,0842,00A4,0308,0470,1805  002D  7525 ;% 04   R[TEMP2]_M[TEMP2].OR.Q                    ; OR opcode with opspecs
U 05. 7700,4868,5BE4,0308,0470,1806  0038  7529 ;% 05   M[TEMP8]_R[TEMP2]                         ; save in M[TEMP8] for error
U 06. 7700,C800,5BE4,03D8,4A70,1807  0043  7533 ;% 06   VA_R[ZERO]                               ; zero the VA register
U 07. 7701,C800,5BE4,0208,5590,1808  004E  7537 ;% 07   WRITE.LONG R[TEMP2]                      ; write data for XB0
U 08. 7700,0800,5BE4,0308,5470,1809  0059  7541 ;% 08   WDR_UNROT(R[TEMP2])                     ; validate cache location 0
U 09. 7700,C800,0364,0300,4470,180A  0064  7545 ;% 09   VA_VA+4                                 ; update VA by 4
U 0A. 7700,3800,4CCD,4DCE,0470,180B  006F  7549 ;% 0A   LONLIT_[66656463]                       ; load opspecs in lonlit reg.
U 0B. 7700,4869,5BE4,03D4,0470,180C  007A  7553 ;% 0B   M[TEMP9]_R[LONLIT]                      ; save in M[TEMP9] for error
U 0C. 7701,8800,5BE4,02D4,5590,180D  0085  7557 ;% 0C   WRITE.LONG R[LONLIT]                    ; write data for XB1
U 0D. 7700,C800,5BE4,03D4,5470,180E  0090  7561 ;% 0D   WDR_UNROT(R[LONLIT])                   ; validate cache location 4
U 0E. 7700,C800,5BE4,03D8,4870,180F  009B  7565 ;% 0E   PC_R[ZERO]                             ; zero PC to force prefetch
U 0F. 7700,C800,0364,1700,0470,1810  00A6  7569 ;% 0F   BUT/IRD1TST                            ; perform ird1
U 10. 7300,4800,0364,0300,0470,1811  00B1  7573 ;% 10   STOP.CLOCK                             ; halt
                                     00BC  7577
                                     00BC  7578 ; IRDX execution to trap generated address
                                     00BC  7579
U 11. 7700,4800,0364,0300,0470,1812  00BC  7580 ;% 11   NOP                                    ; get cpu up to speed
U 12. 7600,8800,0364,0700,0470,1813  00C7  7584 ;% 12   STROBE.V.BUS,BUT/IRDX                  ; do irdx and strobe VBUS
U 13. 7300,4800,0364,0300,0470,1814  00D2  7588 ;% 13   STOP.CLOCK                             ; halt
                                     00DD  7592
                                     00DD  7593 ; FPA enable control and status check
                                     00DD  7594
U 14. 7700,C800,0364,0300,0470,1815  00DD  7595 ;% 14   NOP                                    ; get cpu up to speed
U 15. 5700,C800,0364,0300,2A70,1816  00E8  7599 ;% 15   WCTRL/FPA.ENABLE_WB5,WB_RDM            ; write FPA enable bit
U 16. 7700,4800,0364,0300,0470,1817  00F3  7603 ;% 16   NOP                                    ; ...
U 17. 7700,4800,0364,C300,0470,1800  00FE  7607 ;% 17   BUT/NO.FPA,NEXT/1800                   ; check FPA status
U 18. 7300,C800,0364,0300,0470,1819  0109  7611 ;% 18   STOP.CLOCK                             ; halt
                                     0114  7615
                                     0114  7616 ; Control-P interrupt disable
                                     0114  7617
U 19. 7700,C800,0364,0300,0470,181A  0114  7618 ;% 19   NOP                                    ; get the cpu up to speed
U 1A. 7700,F800,7F9F,FFFF,8470,181B  011F  7622 ;% 1A   LONLIT_[00C00000]                       ; address of s&c bit in crar
U 1B. 7700,4800,5BE4,03D4,2470,181C  012A  7626 ;% 1B   WCTRL/LOADCRAR,WB_R[LONLIT]            ; address the s&c bit
U 1C. 7700,B800,7FDF,FFFF,8470,181D  0135  7630 ;% 1C   LONLIT_[00400000]                       ; data for ctrl-p int. disable
U 1D. 7700,4800,5BE4,03D4,2C70,181E  0140  7634 ;% 1D   WCTRL/CONWRITE,WB_R[LONLIT]            ; disable ctrl-p interrupts
U 1E. 7300,C800,0364,0300,0470,181F  014B  7638 ;% 1E   STOP.CLOCK                             ; halt
                                     0156  7642
                                     0156  7643 ; Control-P interrupt enable
                                     0156  7644
U 1F. 7700,C800,0364,0300,0470,1820  0156  7645 ;% 1F   NOP                                    ; get the cpu up to speed
U 20. 7700,F800,7F9F,FFFF,8470,1821  0161  7649 ;% 20   LONLIT_[00C00000]                       ; address of s&c bit in crar
U 21. 7700,C800,5BE4,03D4,2470,1822  016C  7653 ;% 21   WCTRL/LOADCRAR,WB_R[LONLIT]            ; address the s&c bit
U 22. 7700,7800,7FB7,FFFF,8470,1823  0177  7657 ;% 22   LONLIT_[00900000]                       ; data for ctrl-p int. enable
U 23. 7700,4800,5BE4,03D4,2C70,1824  0182  7661 ;% 23   WCTRL/CONWRITE,WB_R[LONLIT]            ; enable ctrl-p interrupts
U 24. 7300,C800,0364,0300,0470,1825  018D  7665 ;% 24   STOP.CLOCK                             ; halt
                                     0198  7669
                                     0198  7670 ; Set-up RTEMP0 and RTEMP1 for error callout
                                     0198  7671
U 25. 7700,C800,0364,0300,0470,1826  0198  7672 ;% 25   NOP                                    ; get cpu up to speed
U 26. 7700,0848,5924,0300,0470,1827  01A3  7676 ;% 26   R[TEMP0]_M[TEMP8]                      ; move XB0 data to R[TEMP0]
U 27. 7700,0849,5924,0304,0470,1828  01AE  7680 ;% 27   R[TEMP1]_M[TEMP9]                      ; move XB1 data to R[TEMP1]
U 28. 7300,C800,0364,0300,0470,1829  01B9  7684 ;% 28   STOP.CLOCK                             ; halt
                                     01C4  7688
                                     01C4  7689 RTEMP_DATA
                                     0001  7690
```

```
                    00000000  0001  7691          .LONG   0                       ; rtemp0 = contents of xb0
                    00000000  0005  7692          .LONG   0                       ; rtemp1 = contents of xb1
                    00000000  0009  7693          .LONG   0                       ; rtemp2 = instruction
                              000D  7694
                              000D  7695 END_CPU_DATA
                              05DB  7696
                              05DB  7697 END_TEST_DATA
                              05DB
                              05DB        ;-------------------------------------------------------------------------
                              05DB  7698
                              05DB  7699 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
0031          .SBTTL  "TEST   0C8     First IRDX Instruction Address Decode Test 3
0031  7701 ;*****************************************************************************
0031  7702 ;++
0031  7703 ;
0031  7704 ; FUNCTIONAL DESCRIPTION
0031  7705 ;
0031  7706 ;       This test will examine the Native Mode IRDX roms when the IRD Counter
0031  7707 ;       is equal to zero and the operand specifier is for a register reference.
0031  7708 ;
0031  7709 ; TEST ALGORITHM
0031  7710 ;
0031  7711 ;       This test consists of three subtests.  The first subtest will examine
0031  7712 ;       the IRDX roms when the Floating Point Accelerator (FPA) is not present
0031  7713 ;       in the system.  The second subtest will examine the IRDX roms when the
0031  7714 ;       FPA is present but disabled.  The third subtest will examine the IRDX
0031  7715 ;       roms when the FPA is present and enabled.  The algorithm is as follows:
0031  7716 ;
0031  7717 ;        1.  Initialize the processor.
0031  7718 ;        2.  Disable CTRL/P interrupts.
0031  7719 ;        3.  Check to see if FPA present
0031  7720 ;               If FPA present
0031  7721 ;                   Then skip to Subtest 2
0031  7722 ;               Else fall through to subtest 1
0031  7723 ;
0031  7724 ;       Subtest 1
0031  7725 ;
0031  7726 ;        4.  Set-up flag to expect a micro trap.
0031  7727 ;        5.  Initialize loop I for 256 opcodes (00-FF).
0031  7728 ;        6.  Store the index I in a location that will be used to load into the
0031  7729 ;            WDREG.  The value is used as the opcode.
0031  7730 ;        7.  Loop on error from here.
0031  7731 ;        8.  Execute the following test microcode:
0031  7732 ;               a.  NOP to get the RDM up to speed.
0031  7733 ;               b.  Read the WDREG into RTEMP2 (opcode).
0031  7734 ;               c.  Load the operands into the LONLIT register.
0031  7735 ;               d.  Store the operands into the Q register to prepare for ORing
0031  7736 ;                   the opcode with the operands.
0031  7737 ;               e.  OR the opcode and operands together (XB0 data).
0031  7738 ;               f.  Store XB0 data in M[TEMP8] in case needed for error callout.
0031  7739 ;               g.  Zero the VA.
0031  7740 ;               h.  Validate cache location 0 by first writing the data in
0031  7741 ;                   RTEMP2 then reading the data back.
0031  7742 ;               i.  Update the VA by 4.
0031  7743 ;               j.  Load the second operand data into the LONLIT register
0031  7744 ;                   (XB1 data).
0031  7745 ;               k.  Store XB1 data in M[TEMP9] in case needed for error callout.
0031  7746 ;               l.  Validate cache location 4 by first writing the data in
0031  7747 ;                   the LONLIT register then reading the data back.
0031  7748 ;               m.  Load the PC with zero to cause a prefetch to take place.
0031  7749 ;               n.  Do an IRD1 to init the IRD counter and prepare for instruc-
0031  7750 ;                   tion/operand specifier evaluation.
0031  7751 ;               o.  Halt.
0031  7752 ;        9.  Perform the following test microcode.
0031  7753 ;               a.  NOP to get the RDM up to speed.
0031  7754 ;               b.  Do an IRDX.
```

```
0031  7755 ;                   c.  Halt.
0031  7756 ;           10.  Compare the contents of the CSTRP register (output from the IRDX
0031  7757 ;                Roms) against the Rom Table.
0031  7758 ;           11.  If the two do not equal, execute the following microcode and
0031  7759 ;                signal an error (this is done to set-up for printing RTEMP0 & 1
0031  7760 ;                on the error callout):
0031  7761 ;                   a.  NOP to get the RDM up to speed.
0031  7762 ;                   b.  Load RTEMP0 with the contents of M[TEMP8]
0031  7763 ;                   c.  Load RTEMP1 with the contents of M[TEMP9]
0031  7764 ;                   d.  Halt
0031  7765 ;           12.  Continue with loop I for the 256 VAX opcodes.
0031  7766 ;           13.  Re-enable CTRL/P interrupts then skip to next tests.
0031  7767 ;
0031  7768 ;           Subtest 2
0031  7769 ;
0031  7770 ;           14.  Load the FPA disable bit (bit <5> equal to 0).
0031  7771 ;           15.  Loop on error from here.
0031  7772 ;           16.  Set-up to expect micro trap.
0031  7773 ;           17.  Execute the test microcode.  This is done only to get the FPA to
0031  7774 ;                a state where it can be disabled or enabled.  If this had not been
0031  7775 ;                done before then the FPA would be setting in the initialize state
0031  7776 ;                which forces FPA enable.
0031  7777 ;           18.  Execute the following microcode to disable the FPA:
0031  7778 ;                   a.  NOP to get the RDM up to speed.
0031  7779 ;                   b.  Read the RDM WDREG onto the W Bus and then write the W Bus
0031  7780 ;                       to the FPA.
0031  7781 ;                   c.  Perform a NO.FPA test to insure that the FPA got disabled.
0031  7782 ;                   d.  Halt.
0031  7783 ;           19.  Check the Trap register to insure that the FPA got disabled.
0031  7784 ;           20.  If the FPA is not disabled, then signal an error.
0031  7785 ;           21.  Set-up flag to expect a micro trap.
0031  7786 ;           22.  Initialize loop I for 256 opcodes (00-FF).
0031  7787 ;           23.  Store the index I in a location that will be used to load into the
0031  7788 ;                WDREG.  The value is used as the opcode.
0031  7789 ;           24.  Loop on error from here.
0031  7790 ;           25.  Execute the following test microcode:
0031  7791 ;                   a.  NOP to get the RDM up to speed.
0031  7792 ;                   b.  Read the WDREG into RTEMP2 (opcode).
0031  7793 ;                   c.  Load the operands into the LONLIT register.
0031  7794 ;                   d.  Store the operands into the Q register to prepare for ORing
0031  7795 ;                       the opcode with the operands.
0031  7796 ;                   e.  OR the opcode and operands together (XB0 data).
0031  7797 ;                   f.  Store XB0 data in M[TEMP8] in case needed for error callout.
0031  7798 ;                   g.  Zero the VA.
0031  7799 ;                   h.  Validate cache location 0 by first writing the data in
0031  7800 ;                       RTEMP2 then reading the data back.
0031  7801 ;                   i.  Update the VA by 4.
0031  7802 ;                   j.  Load the second operand data into the LONLIT register
0031  7803 ;                       (XB1 data).
0031  7804 ;                   k.  Store XB1 data in M[TEMP9] in case needed for error callout.
0031  7805 ;                   l.  Validate cache location 4 by first writing the data in
0031  7806 ;                       the LONLIT register then reading the data back.
0031  7807 ;                   m.  Load the PC with zero to cause a prefetch to take place.
0031  7808 ;                   n.  Do an IRD1 to init the IRD counter and prepare for instruc-
0031  7809 ;                       tion/operand specifier evaluation.
```

```
0031  7810 ;                   o.  Halt.
0031  7811 ;           26.  Perform the following test microcode.
0031  7812 ;                   a.  NOP to get the RDM up to speed.
0031  7813 ;                   b.  Do an IRDX.
0031  7814 ;                   c.  Halt.
0031  7815 ;           27.  Compare the contents of the CSTRP register (output from the IRDX
0031  7816 ;                Roms) against the Rom Table.
0031  7817 ;           28.  If the two do not equal, execute the following microcode and
0031  7818 ;                signal an error (this is done to set-up for printing RTEMP0 & 1
0031  7819 ;                on the error callout):
0031  7820 ;                   a.  NOP to get the RDM up to speed.
0031  7821 ;                   b.  Load RTEMP0 with the contents of M[TEMP8]
0031  7822 ;                   c.  Load RTEMP1 with the contents of M[TEMP9]
0031  7823 ;                   d.  Halt
0031  7824 ;           29.  Continue with loop I for the 256 VAX opcodes.
0031  7825 ;           30.  Go to Subtest 3.
0031  7826 ;
0031  7827 ;           Subtest 3
0031  7828 ;
0031  7829 ;           31.  Load the FPA enable bit (bit <5> equal to 1).
0031  7830 ;           32.  Loop on error from here.
0031  7831 ;           33.  Set-up to expect micro trap.
0031  7832 ;           34.  Execute the following microcode to enable the FPA:
0031  7833 ;                   a.  NOP to get the RDM up to speed.
0031  7834 ;                   b.  Read the RDM WDREG onto the W Bus and then write the W Bus
0031  7835 ;                       to the FPA.
0031  7836 ;                   c.  Perform a NO.FPA test to insure that the FPA got enabled.
0031  7837 ;                   d.  Halt.
0031  7838 ;           35.  Check the Trap register to insure that the FPA got enabled.
0031  7839 ;           36.  If the FPA is not enabled, then signal an error.
0031  7840 ;           37.  Set-up flag to expect a micro trap.
0031  7841 ;           38.  Initialize loop I for 256 opcodes (00-FF).
0031  7842 ;           39.  Store the index I in a location that will be used to load into the
0031  7843 ;                WDREG.  The value is used as the opcode.
0031  7844 ;           40.  Loop on error from here.
0031  7845 ;           41.  Set then Clear the DCLO bit in the RDCTRL register to perform an
0031  7846 ;                initialize.  This is done because when the test microcode
0031  7847 ;                executes, if the opcode was an FP opcode then there is a
0031  7848 ;                possibility that the processor could be setting there in an
0031  7849 ;                FPAWAIT state (M clock and D clock stalled).  If this were to
0031  7850 ;                happen then the cpu clocks would no longer run and the remaining
0031  7851 ;                test opcodes would fail.
0031  7852 ;           42.  Execute the following test microcode:
0031  7853 ;                   a.  NOP to get the RDM up to speed.
0031  7854 ;                   b.  Read the WDREG into RTEMP2 (opcode).
0031  7855 ;                   c.  Load the operands into the LONLIT register.
0031  7856 ;                   d.  Store the operands into the Q register to prepare for ORing
0031  7857 ;                       the opcode with the operands.
0031  7858 ;                   e.  OR the opcode and operands together (XB0 data).
0031  7859 ;                   f.  Store XB0 data in M[TEMP8] in case needed for error callout.
0031  7860 ;                   g.  Zero the VA.
0031  7861 ;                   h.  Validate cache location 0 by first writing the data in
0031  7862 ;                       RTEMP2 then reading the data back.
0031  7863 ;                   i.  Update the VA by 4.
0031  7864 ;                   j.  Load the second operand data into the LONLIT register
```

```
0031  7865 ;                    (XB1 data).
0031  7866 ;                k.  Store XB1 data in M[TEMP9] in case needed for error callout.
0031  7867 ;                l.  Validate cache location 4 by first writing the data in
0031  7868 ;                    the LONLIT register then reading the data back.
0031  7869 ;                m.  Load the PC with zero to cause a prefetch to take place.
0031  7870 ;                n.  Do an IRD1 to init the IRD counter and prepare for instruc-
0031  7871 ;                    tion/operand specifier evaluation.
0031  7872 ;                o.  Halt.
0031  7873 ;           43.  Perform the following test microcode.
0031  7874 ;                a.  NOP to get the RDM up to speed.
0031  7875 ;                b.  Do an IRDX.
0031  7876 ;                c.  Halt.
0031  7877 ;           44.  Compare the contents of the CSTRP register (output from the IRDX
0031  7878 ;                Roms) against the Rom Table.
0031  7879 ;           45.  If the two do not equal, execute the following microcode and
0031  7880 ;                signal an error (this is done to set-up for printing RTEMP0 & 1
0031  7881 ;                on the error callout):
0031  7882 ;                a.  NOP to get the RDM up to speed.
0031  7883 ;                b.  Load RTEMP0 with the contents of M[TEMP8]
0031  7884 ;                c.  Load RTEMP1 with the contents of M[TEMP9]
0031  7885 ;                d.  Halt
0031  7886 ;           46.  Continue with loop I for the 256 VAX opcodes.
0031  7887 ;           47.  Re-enable CTRL/P interrupts.
0031  7888 ;           48.  Skip to next tests.
0031  7889 ;
0031  7890 ; TEST PATTERNS
0031  7891 ;
0031  7892 ;           For this test the opcodes are generated by a simple count pattern.  The
0031  7893 ;           opcodes are based on the 256 base machine instructions as specified in
0031  7894 ;           the VAX architecture handbook.  The generated opcodes are written to
0031  7895 ;           the CPU and combined (ORed) with three bytes of operand data to form
0031  7896 ;           a machine instruction.  The operands all specify Register mode using
0031  7897 ;           registers 0-6 and are loaded from the LONLIT register.
0031  7898 ;
0031  7899 ;           Also required are two tables of expected control store addresses for
0031  7900 ;           the first IRDX cycle.  One table contains the expected addresses for
0031  7901 ;           the first IRDX when the FPA is disabled, the other for when it is
0031  7902 ;           enabled.
0031  7903 ;
0031  7904 ; LOGIC DESCRIPTION
0031  7905 ;
0031  7906 ;           This test will check the validity of the IRDX roms on the DPM module.
0031  7907 ;           The IRDX roms are addressed by bits <7:0> of the Instruction Register.
0031  7908 ;           Along with being addressed by the IR, there are three other signals
0031  7909 ;           that affect the address.  They are IRD CTR 0 H, FPA ENABLED L, and
0031  7910 ;           REG MODE H.  This test will examine every 8th(hex) address starting at
0031  7911 ;           address 3 with FPA disabled and starting at address 1 with FPA enabled.
0031  7912 ;
0031  7913 ; ASSUMPTIONS
0031  7914 ;
0031  7915 ;           This test assumes the following logic has already been tested; the
0031  7916 ;           CPU's data path, the cache, the XB, and the IR and OSR.
0031  7917 ;
0031  7918 ; ERROR DESCRIPTION
0031  7919 ;
```

```
                     0031  7920 ;       Subtest 1
                     0031  7921 ;
                     0031  7922 ;               Expected CS Address
                     0031  7923 ;               Received CS Address
                     0031  7924 ;
                     0031  7925 ;       Subtest 2
                     0031  7926 ;
                     0031  7927 ;               First IFERROR
                     0031  7928 ;
                     0031  7929 ;                       Expected CS Address
                     0031  7930 ;                       Received CS Address
                     0031  7931 ;
                     0031  7932 ;               Second IFERROR
                     0031  7933 ;
                     0031  7934 ;                       Expected IRDX Rom Contents
                     0031  7935 ;                       Received IRDX Rom Contents
                     0031  7936 ;                       Loop I count
                     0031  7937 ;                       Contents of XB0
                     0031  7938 ;                       Contents of XB1
                     0031  7939 ;
                     0031  7940 ;       Subtest 3
                     0031  7941 ;
                     0031  7942 ;               First IFERROR
                     0031  7943 ;
                     0031  7944 ;                       Expected CS Address
                     0031  7945 ;                       Received CS Address
                     0031  7946 ;
                     0031  7947 ;               Second IFERROR
                     0031  7948 ;
                     0031  7949 ;                       Expected IRDX Rom Contents
                     0031  7950 ;                       Received IRDX Rom Contents
                     0031  7951 ;                       Loop I count
                     0031  7952 ;                       Contents of XB0
                     0031  7953 ;                       Contents of XB1
                     0031  7954 ;--
                     0031  7955 ;*******************************************************************************
                     0031  7956
                     0031  7957         INITIALIZE                                  ; initialize the comet cpu
                     0033  7958         FETCH           <^X19>                      ; disable ctrl-p interrupts
                     0038  7959         BRSTCLK         <^X1E>                      ; ...
                     003C  7960         FETCH           <^X16>                      ; execute a but/no.fpa to check
                     0041  7961         BRSTCLK         <^X18>                      ; for fpa present
                     0045  7962         CMPREGMSK       CSTRP,1$,,2$,,W             ; is the fpa present?
                     0051  7963         SKIP            100$,ONERROR                ; if so then skip to subtest 2
                     0058  7964
                     0058  7965         SUBTEST         1
                     005B
                     005B               ;////////////////////////////////////////////////////////////////////////////
                     005B  7966 ;*
                     005B  7967 ;
                     005B  7968 ; FPA is not present.
                     005B  7969 ;
                     005B  7970 ;-
                     005B  7971         EXPUTRAP                                    ; else expect non-dcs addresses
                     005D  7972         LOOP            I,1,256                     ; loop for 256 opcodes
```

```
                        0064  7973          SAVEINDEX       I,9$                    ; save the loop count
                        0069  7974          LOADREG         WDREG,9$,,L             ; load wdreg with loop count
                        0071  7975          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        0073  7976          FETCH           0                       ; load the istrm data and do an
                        0078  7977          BRSTCLK         <^X10>,INH              ; ird1tst
                        007C  7978          FETCH           <^X11>                  ; execute the irdx
                        0081  7979          BRSTCLK         <^X13>,INH              ; ...
                        0085  7980          CMPREGMSK       CSTRP,30$,I,6$,,W       ; test the cs address
                        0091  7981          SKIP            50$,NOERROR             ; continue testing if no error
                        0098  7982          FETCH           <^X25>                  ; start dcs program
                        009D  7983          BRSTCLK         <^X28>                  ; end dcs program
                        00A1  7984          IFERROR         2,<<IRD>>,<<MIC>>       ; if bad address, signal error
                        00A8  7985 50$:     ENDLOOP         I                       ; continue loop
                        00AB  7986          FETCH           <^X1F>                  ; re-enable ctrl-p interrupts
                        00B0  7987          BRSTCLK         <^X24>                  ; ...
                        00B4  7988          SKIP                                    ; skip on to the next test
                        00BB  7989
                        00BB  7990 100$:    SUBTEST         2
                        00BE
                        00BE        ;////////////////////////////////////////////////////////////////////////////////
                        00BE  7991 ;*
                        00BE  7992 ;
                        00BE  7993 ; FPA present but not enabled
                        00BE  7994 ;
                        00BE  7995 ;-
                        00BE  7996          LOADREG         WDREG,7$,,L             ; load fpa disable bit
                        00C6  7997          ERRLOOP                                 ; loop on error here
                        00C8  7998          EXPUTRAP                                ; expect micro trap
                        00CA  7999          FETCH           0                       ; start dcs program
                        00CF  8000          BRSTCLK         <^X10>,INH              ; end dcs program
                        00D3  8001          FETCH           <^X14>                  ; execute the disable
                        00D8  8002          BRSTCLK         <^X18>,INH              ; and read back fpa status
                        00DC  8003          CMPREGMSK       CSTRP,3$,,5$,,W         ; is the fpa disabled?
                        00E8  8004          IFERROR         ,,<FPA>                 ; if not, signal an error
                        00EE  8005          EXPUTRAP                                ; else expect non-dcs addresses
                        00F0  8006          LOOP            I,1,256                 ; loop for 256 opcodes
                        00F7  8007          SAVEINDEX       I,9$                    ; save the loop count
                        00FC  8008          LOADREG         WDREG,9$,,L             ; load wdreg with loop count
                        0104  8009          ERRLOOP                                 ; Loop on error from here
                        0106  8010          FETCH           0                       ; execute a but/ird1tst
                        010B  8011          BRSTCLK         <^X10>,INH              ; ...
                        010F  8012          FETCH           <^X11>                  ; execute an irdx
                        0114  8013          BRSTCLK         <^X13>,INH              ; ...
                        0118  8014          CMPREGMSK       CSTRP,30$,I,6$,,W       ; is the cs address correct?
                        0124  8015          SKIP            51$,NOERROR             ; continue testing if no error
                        012B  8016          FETCH           <^X25>                  ; start dcs program
                        0130  8017          BRSTCLK         <^X28>                  ; end dcs program
                        0134  8018          IFERROR         2,<<IRD>>,<<MIC>>       ; if error then signal an error
                        013B  8019 51$:     ENDLOOP         I                       ; continue loop
                        013E  8020
                        013E  8021          SUBTEST         3
                        0141
                        0141        ;////////////////////////////////////////////////////////////////////////////////
                        0141  8022 ;*
                        0141  8023 ;
```

```
                        0141  8024 ; FPA present and enabled
                        0141  8025 ;
                        0141  8026 ;-
                        0141  8027          LOADREG         WDREG,8$,,L             ; load fpa enable bit
                        0149  8028          ERRLOOP                                 ; loop on error from here
                        014B  8029          FETCH           <^X14>                  ; enable the fpa
                        0150  8030          BRSTCLK         <^X18>,INH              ; and read back fpa enable
                        0154  8031          CMPREGMSK       CSTRP,4$,,5$,,W         ; is the fpa enabled?
                        0160  8032          IFERROR         ,,<FPA>                 ; if not, signal an error
                        0166  8033          EXPUTRAP                                ; else expect non-dcs addresses
                        0168  8034          LOOP            I,1,256                 ; loop for 256 opcodes
                        016F  8035          SAVEINDEX       I,9$                    ; save the loop count
                        0174  8036          LOADREG         WDREG,9$,,L             ; load wdreg with loop count
                        017C  8037          ERRLOOP                                 ; loop on error from here
                        017E  8038          LOADREG         RDCTRL,10$,,B           ; assert dclo
                        0186  8039          LOADREG         RDCTRL,11$,,B           ; de-assert dclo
                        018E  8040          FETCH           0                       ; execute a but/ird1tst
                        0193  8041          BRSTCLK         <^X10>,INH              ; ...
                        0197  8042          FETCH           <^X11>                  ; execute the but/irdx
                        019C  8043          BRSTCLK         <^X13>,INH              ; ...
                        01A0  8044          CMPREGMSK       CSTRP,31$,I,6$,,W       ; is the cs address correct?
                        01AC  8045          SKIP            52$,NOERROR             ; continue testing if no error
                        01B3  8046          FETCH           <^X25>                  ; start dcs program
                        01B8  8047          BRSTCLK         <^X28>                  ; end dcs program
                        01BC  8048          IFERROR         2,<<IRD>>,<<MIC>>       ; signal an error
                        01C3  8049 52$:     ENDLOOP         I                       ; continue loop
                        01C6  8050          FETCH           <^X1F>                  ; re-enable ctrl-p interrupts
                        01CB  8051          BRSTCLK         <^X24>                  ; ...
                        01CF  8052          SKIP                                    ; end of test
                        01D6  8053
                        01D6  8054 BEGIN_TEST_DATA
                        01D6
                        01D6       ;-----------------------------------------------------------------------------
                        01D6  8055
                1802    01D6  8056 1$:      .WORD   ^X1802
                3FFE    01D8  8057 2$:      .WORD   ^X3FFE
                        01DA  8058
                1801    01DA  8059 3$:      .WORD   ^X1801
                1800    01DC  8060 4$:      .WORD   ^X1800
                3FFD    01DE  8061 5$:      .WORD   ^X3FFD
                        01E0  8062
                3FFF    01E0  8063 6$:      .WORD   ^X3FFF
                        01E2  8064
            00000000    01E2  8065 7$:      .LONG   0
            00000020    01E6  8066 8$:      .LONG   ^X00000020
                        01EA  8067
            00000000    01EA  8068 9$:      .LONG   0
                  40    01EE  8069 10$:     .BYTE   DC_LOW
                  00    01EF  8070 11$:     .BYTE   ^X00
                        01F0  8071
                        01F0  8072 ;
                        01F0  8073 ; Expected CS Addresses for Subtests 1 and 2
                        01F0  8074 ;
                        01F0  8075 ; The following is a table of all the expected control store addresses for the
                        01F0  8076 ; first IRDX.
```

```
                                        01F0  8077 ;
                                        01F0  8078
03F8 03F8 03F8 03F8 03F8 03F8 03F8      01F0  8079 30$:     .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 00 - 06
0100 0100 0381 0100 03BF 03BE 03F8      01FE  8080          .WORD   ^X03F8,^X03BE,^X03BF,^X0100,^X0381,^X0100,^X0100 ; 07 - 0D
03F8 03F8 03F8 03F8 03F8 03AB 0120      020C  8081          .WORD   ^X0120,^X03AB,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 0E - 14
03F8 03F8 03F8 03F8 0396 03A4 03F8      021A  8082          .WORD   ^X03F8,^X03A4,^X0396,^X03F8,^X03F8,^X03F8,^X03F8 ; 15 - 1B
03B9 0288 03B9 03F8 03F8 03F8 03F8      0228  8083          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03B9,^X0288,^X03B9 ; 1C - 22
03AE 03AE 0288 03BD 0288 03BC 0288      0236  8084          .WORD   ^X0288,^X03BC,^X0288,^X03BD,^X0288,^X03AE,^X03AE ; 23 - 29
03F8 03B2 03B2 03AF 03AF 03B4 03B4      0244  8085          .WORD   ^X03B4,^X03B4,^X03AF,^X03AF,^X03B2,^X03B2,^X03F8 ; 2A - 30
03B9 03BA 03B7 03B7 00CE 00E5 03F8      0252  8086          .WORD   ^X03F8,^X00E5,^X00CE,^X03B7,^X03B7,^X03BA,^X03B9 ; 31 - 37
00F1 0100 01DE 03B5 03B5 03B6 03C1      0260  8087          .WORD   ^X03C1,^X03B6,^X03B5,^X03B5,^X01DE,^X0100,^X00F1 ; 38 - 3E
0160 0110 0349 0345 0160 0110 0206      026E  8088          .WORD   ^X0206,^X0110,^X0160,^X0345,^X0349,^X0110,^X0160 ; 3F - 45
00B2 0150 0150 0150 0150 0160 0110      027C  8089          .WORD   ^X0110,^X0160,^X0150,^X0150,^X0150,^X0150,^X00B2 ; 46 - 4C
0329 0325 0160 0311 0160 0150 00B2      028A  8090          .WORD   ^X00B2,^X0150,^X0160,^X0311,^X0160,^X0325,^X0329 ; 4D - 53
03F8 03F8 0247 03F8 001F 0100 0100      0298  8091          .WORD   ^X0100,^X0100,^X001F,^X03F8,^X0247,^X03F8,^X03F8 ; 54 - 5A
00B4 00A8 03AD 03AC 0120 0120 03F8      02A6  8092          .WORD   ^X03F8,^X0120,^X0120,^X03AC,^X03AD,^X00A8,^X00B4 ; 5B - 61
0068 00D8 00D4 00C8 00B8 0357 0356      02B4  8093          .WORD   ^X0356,^X0357,^X00B8,^X00C8,^X00D4,^X00D8,^X0068 ; 62 - 68
0359 00B2 00B2 00B2 0074 0068 0068      02C2  8094          .WORD   ^X0068,^X0068,^X0074,^X00B2,^X00B2,^X00B2,^X0359 ; 69 - 6F
0150 035A 00D2 032D 0062 0331 0060      02D0  8095          .WORD   ^X0060,^X0331,^X0062,^X032D,^X00D2,^X035A,^X0150 ; 70 - 76
01E5 022A 0170 0100 0170 0100 03F8      02DE  8096          .WORD   ^X03F8,^X0100,^X0170,^X0100,^X0170,^X022A,^X01E5 ; 77 - 7D
0110 0100 0251 0100 0237 020D 01F1      02EC  8097          .WORD   ^X01F1,^X020D,^X0237,^X0100,^X0251,^X0100,^X0110 ; 7E - 84
0100 028E 0100 027F 02CD 02B9 0100      02FA  8098          .WORD   ^X0100,^X02B9,^X02CD,^X027F,^X0100,^X028E,^X0100 ; 85 - 8B
01BF 006E 00F1 0100 01D0 0100 0277      0308  8099          .WORD   ^X0277,^X0100,^X01D0,^X0100,^X00F1,^X006E,^X01BF ; 8C - 92
00E5 00E5 0265 0243 008E 0221 004E      0316  8100          .WORD   ^X004E,^X0221,^X008E,^X0243,^X0265,^X00E5,^X00E5 ; 93 - 99
0237 0206 00F1 0100 0100 01DE 01DE      0324  8101          .WORD   ^X01DE,^X01DE,^X0100,^X0100,^X00F1,^X0206,^X0237 ; 9A - A0
0301 02BF 0100 0110 0100 0251 0100      0332  8102          .WORD   ^X0100,^X0251,^X0100,^X0110,^X0100,^X02BF,^X0301 ; A1 - A7
01D0 0100 0277 0100 028E 0100 027F      0340  8103          .WORD   ^X027F,^X0100,^X028E,^X0100,^X0277,^X0100,^X01D0 ; A8 - AE
008E 0221 004E 01BF 006E 00F1 0100      034E  8104          .WORD   ^X0100,^X00F1,^X006E,^X01BF,^X004E,^X0221,^X008E ; AF - B5
03C6 03A6 03A7 029E 0292 0265 0243      035C  8105          .WORD   ^X0243,^X0265,^X0292,^X029E,^X03A7,^X03A6,^X03C6 ; B6 - BC
0100 0251 0100 0237 03C9 03C8 03C7      036A  8106          .WORD   ^X03C7,^X03C8,^X03C9,^X0237,^X0100,^X0251,^X0100 ; BD - C3
028E 0100 027F 0305 02C9 0100 0110      0378  8107          .WORD   ^X0110,^X0100,^X02C9,^X0305,^X027F,^X0100,^X028E ; C4 - CA
006E 00F1 0100 01D0 0100 0277 0100      0386  8108          .WORD   ^X0100,^X0277,^X0100,^X01D0,^X0100,^X00F1,^X006E ; CB - D1
022E 0265 0243 008E 0221 004E 01BF      0394  8109          .WORD   ^X01BF,^X004E,^X0221,^X008E,^X0243,^X0265,^X022E ; D2 - D8
0206 00F1 0206 03A9 0398 0100 026E      03A2  8110          .WORD   ^X026E,^X0100,^X0398,^X03A9,^X0206,^X00F1,^X0206 ; D9 - DF
0397 0397 0397 0397 0397 0397 0397      03B0  8111          .WORD   ^X0397,^X0397,^X0397,^X0397,^X0397,^X0397,^X0397 ; E0 - E6
0100 0100 0100 0100 039E 039D 0397      03BE  8112          .WORD   ^X0397,^X039D,^X039E,^X0100,^X0100,^X0100,^X0100 ; E7 - ED
03A2 03A1 03A1 0100 0100 0100 0100      03CC  8113          .WORD   ^X0100,^X0100,^X0100,^X0100,^X03A1,^X03A1,^X03A2 ; EE - F4
0120 0120 0100 0326 00CE 00CE 03A2      03DA  8114          .WORD   ^X03A2,^X00CE,^X00CE,^X0326,^X0100,^X0120,^X0120 ; F5 - FB
               03F8 03F8 0776 0776      03E8  8115          .WORD   ^X0776,^X0776,^X03F8,^X03F8                       ; FC - FF
                                        03F0  8116
                                        03F0  8117 ;
                                        03F0  8118 ; Expected CS Addresses for Subtest 3
                                        03F0  8119 ;
                                        03F0  8120
03F8 03F8 03F8 03F8 03F8 03F8 03F8      03F0  8121 31$:     .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 00 - 06
0100 0100 0381 0100 03BF 03BE 03F8      03FE  8122          .WORD   ^X03F8,^X03BE,^X03BF,^X0100,^X0381,^X0100,^X0100 ; 07 - 0D
03F8 03F8 03F8 03F8 03F8 03AB 0120      040C  8123          .WORD   ^X0120,^X03AB,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 0E - 14
03F8 03F8 03F8 03F8 0396 03A4 03F8      041A  8124          .WORD   ^X03F8,^X03A4,^X0396,^X03F8,^X03F8,^X03F8,^X03F8 ; 15 - 1B
03B9 0288 03B9 03F8 03F8 03F8 03F8      0428  8125          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03B9,^X0288,^X03B9 ; 1C - 22
03AE 03AE 0288 03BD 0288 03BC 0288      0436  8126          .WORD   ^X0288,^X03BC,^X0288,^X03BD,^X0288,^X03AE,^X03AE ; 23 - 29
03F8 03B2 03B2 03AF 03AF 03B4 03B4      0444  8127          .WORD   ^X03B4,^X03B4,^X03AF,^X03AF,^X03B2,^X03B2,^X03F8 ; 2A - 30
03B9 03BA 03B7 03B7 00CE 00E5 03F8      0452  8128          .WORD   ^X03F8,^X00E5,^X00CE,^X03B7,^X03B7,^X03BA,^X03B9 ; 31 - 37
00F1 0100 01DE 03B5 03B5 03B6 03C1      0460  8129          .WORD   ^X03C1,^X03B6,^X03B5,^X03B5,^X01DE,^X0100,^X00F1 ; 38 - 3E
00DC 035C 00DC 035C 00DC 035C 0206      046E  8130          .WORD   ^X0206,^X035C,^X00DC,^X035C,^X00DC,^X035C,^X00DC ; 3F - 45
00E6 00E6 00E6 00E6 00E6 00DC 035C      047C  8131          .WORD   ^X035C,^X00DC,^X00E6,^X00E6,^X00E6,^X00E6,^X00E6 ; 46 - 4C
```

```
                                                
```

```
U 13, 7300,4800,0364,0300,0470,1814  00D2  8244 ;% 13   STOP.CLOCK                          ; halt
                                     00DD  8248
                                     00DD  8249 ; FPA enable control and status check
                                     00DD  8250
U 14, 7700,C800,0364,0300,0470,1815  00DD  8251 ;% 14   NOP                                  ; get cpu up to speed
U 15, 5700,C800,0364,0300,2A70,1816  00E8  8255 ;% 15   WCTRL/FPA.ENABLE_WB5,WB_RDM          ; write the FPA enable bit
U 16, 7700,4800,0364,0300,0470,1817  00F3  8259 ;% 16   NOP                                  ; ...
U 17, 7700,4800,0364,C300,0470,1800  00FE  8263 ;% 17   BUT/NO.FPA,NEXT/1800                ; check FPA status
U 18, 7300,C800,0364,0300,0470,1819  0109  8267 ;% 18   STOP.CLOCK                          ; halt
                                     0114  8271
                                     0114  8272 ; Control-P interrupt disable
                                     0114  8273
U 19, 7700,C800,0364,0300,0470,181A  0114  8274 ;% 19   NOP                                  ; get the cpu up to speed
U 1A, 7700,F800,7F9F,FFFF,8470,181B  011F  8278 ;% 1A   LONLIT_[00C00000]                   ; address of s&c bit in crar
U 1B, 7700,4800,5BE4,03D4,2470,181C  012A  8282 ;% 1B   WCTRL/LOADCRAR,WB_R[LONLIT]          ; address the s&c bit
U 1C, 7700,B800,7FDF,FFFF,8470,181D  0135  8286 ;% 1C   LONLIT_[00400000]                   ; data for ctrl-p int. disable
U 1D, 7700,4800,5BE4,03D4,2C70,181E  0140  8290 ;% 1D   WCTRL/CONWRITE,WB_R[LONLIT]          ; disable ctrl-p interrupts
U 1E, 7300,C800,0364,0300,0470,181F  014B  8294 ;% 1E   STOP.CLOCK                          ; halt
                                     0156  8298
                                     0156  8299 ; Control-P interrupt enable
                                     0156  8300
U 1F, 7700,C800,0364,0300,0470,1820  0156  8301 ;% 1F   NOP                                  ; get the cpu up to speed
U 20, 7700,F800,7F9F,FFFF,8470,1821  0161  8305 ;% 20   LONLIT_[00C00000]                   ; address of s&c bit in crar
U 21, 7700,C800,5BE4,03D4,2470,1822  016C  8309 ;% 21   WCTRL/LOADCRAR,WB_R[LONLIT]          ; address the s&c bit
U 22, 7700,7800,7FB7,FFFF,8470,1823  0177  8313 ;% 22   LONLIT_[00900000]                   ; data for ctrl-p int. enable
U 23, 7700,4800,5BE4,03D4,2C70,1824  0182  8317 ;% 23   WCTRL/CONWRITE,WB_R[LONLIT]          ; enable ctrl-p interrupts
U 24, 7300,C800,0364,0300,0470,1825  018D  8321 ;% 24   STOP.CLOCK                          ; halt
                                     0198  8325
                                     0198  8326 ; Set-up RTEMP0 and RTEMP1 for error callout
                                     0198  8327
U 25, 7700,C800,0364,0300,0470,1826  0198  8328 ;% 25   NOP                                  ; get the cpu up to speed
U 26, 7700,0848,5924,0300,0470,1827  01A3  8332 ;% 26   R[TEMP0]_M[TEMP8]                   ; move XB0 data to R[TEMP0]
U 27, 7700,0849,5924,0304,0470,1828  01AE  8336 ;% 27   R[TEMP1]_M[TEMP9]                   ; move XB1 data to R[TEMP1]
U 28, 7300,C800,0364,0300,0470,1829  01B9  8340 ;% 28   STOP.CLOCK                          ; halt
                                     01C4  8344
                                     01C4  8345 RTEMP_DATA
                                     0001  8346
                          00000000   0001  8347         .LONG   0                    ; rtemp0 = contents of xb0
                          00000000   0005  8348         .LONG   0                    ; rtemp1 = contents of xb1
                          00000000   0009  8349         .LONG   0                    ; rtemp2 = instruction
                                     000D  8350
                                     000D  8351 END_CPU_DATA
                                     05F0  8352
                                     05F0  8353 END_TEST_DATA
                                     05F0
                                     05F0       ;-----------------------------------------------------------------------
                                     05F0  8354
                                     05F0  8355 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
0031          .SBTTL  "TEST   0C9     First IRDX Instruction Address Decode Test 4
0031   8357 ;*****************************************************************************
0031   8358 ;++
0031   8359 ;
0031   8360 ; FUNCTIONAL DESCRIPTION
0031   8361 ;
0031   8362 ;       This test will examine the signal ROM OS INH H from the Native Mode
0031   8363 ;       IRDX roms when the IRD Counter is equal to zero and the operand
0031   8364 ;       specifier is for a register reference.
0031   8365 ;
0031   8366 ; TEST ALGORITHM
0031   8367 ;
0031   8368 ;       This test consists of three subtests.  The first subtest will examine
0031   8369 ;       the IRDX roms when the Floating Point Accelerator (FPA) is not present
0031   8370 ;       in the system.  The second subtest will examine the IRDX roms when the
0031   8371 ;       FPA is present but disabled.  The third subtest will examine the IRDX
0031   8372 ;       roms when the FPA is present and enabled.  The algorithm is as follows:
0031   8373 ;
0031   8374 ;        1.  Initialize the processor.
0031   8375 ;        2.  Disable CTRL/P interrupts.
0031   8376 ;        3.  Check to see if FPA present
0031   8377 ;               If FPA present
0031   8378 ;                   Then skip to Subtest 2
0031   8379 ;               Else fall through to subtest 1
0031   8380 ;
0031   8381 ;       Subtest 1
0031   8382 ;
0031   8383 ;        4.  Set-up flag to expect a micro trap.
0031   8384 ;        5.  Initialize loop I for 256 opcodes (00-FF).
0031   8385 ;        6.  Store the index I in a location that will be used to load into the
0031   8386 ;            WDREG.  The value is used as the opcode.
0031   8387 ;        7.  Loop on error from here.
0031   8388 ;        8.  Execute the following test microcode:
0031   8389 ;               a.  NOP to get the RDM up to speed.
0031   8390 ;               b.  Read the WDREG into RTEMP2 (opcode).
0031   8391 ;               c.  Load the operands into the LONLIT register.
0031   8392 ;               d.  Store the operands into the Q register to prepare for ORing
0031   8393 ;                   the opcode with the operands.
0031   8394 ;               e.  OR the opcode and operands together (XB0 data).
0031   8395 ;               f.  Store XB0 data in M[TEMP8] in case needed for error callout.
0031   8396 ;               g.  Zero the VA.
0031   8397 ;               h.  Validate cache location 0 by first writing the data in
0031   8398 ;                   RTEMP2 then reading the data back.
0031   8399 ;               i.  Update the VA by 4.
0031   8400 ;               j.  Load the second operand data into the LONLIT register
0031   8401 ;                   (XB1 data).
0031   8402 ;               k.  Store XB1 data in M[TEMP9] in case needed for error callout.
0031   8403 ;               l.  Validate cache location 4 by first writing the data in
0031   8404 ;                   the LONLIT register then reading the data back.
0031   8405 ;               m.  Load the PC with zero to cause a prefetch to take place.
0031   8406 ;               n.  Do an IRD1 to init the IRD counter and prepare for instruc-
0031   8407 ;                   tion/operand specifier evaluation.
0031   8408 ;               o.  Halt.
0031   8409 ;        9.  Perform the following test microcode.
0031   8410 ;               a.  NOP to get the RDM up to speed.
```

```
0031  8411 ;                  b.  Do an IRDX and strobe the VBUS.
0031  8412 ;                  c.  Halt.
0031  8413 ;          10.  Compare the VBUS data with the expected value and bit position
0031  8414 ;               from the VBUS Table.
0031  8415 ;          11.  If the two do not equal, execute the following microcode and
0031  8416 ;               signal an error (this is done to set-up for printing RTEMP0 & 1
0031  8417 ;               on the error callout):
0031  8418 ;                  a.  NOP to get the RDM up to speed.
0031  8419 ;                  b.  Load RTEMP0 with the contents of M[TEMP8]
0031  8420 ;                  c.  Load RTEMP1 with the contents of M[TEMP9]
0031  8421 ;                  d.  Halt
0031  8422 ;          12.  Continue with loop I for the 256 VAX opcodes.
0031  8423 ;          13.  Re-enable CTRL/P interrupts then skip to next tests.
0031  8424 ;
0031  8425 ;          Subtest 2
0031  8426 ;
0031  8427 ;          14.  Load the FPA disable bit (bit <5> equal to 0).
0031  8428 ;          15.  Loop on error from here.
0031  8429 ;          16.  Set-up to expect micro trap.
0031  8430 ;          17.  Execute the test microcode.  This is done only to get the FPA to
0031  8431 ;               a state where it can be disabled or enabled.  If this had not been
0031  8432 ;               done before then the FPA would be setting in the initialize state
0031  8433 ;               which forces FPA enable.
0031  8434 ;          18.  Execute the following microcode to disable the FPA:
0031  8435 ;                  a.  NOP to get the RDM up to speed.
0031  8436 ;                  b.  Read the RDM WDREG onto the W Bus and then write the W Bus
0031  8437 ;                      to the FPA.
0031  8438 ;                  c.  Perform a NO.FPA test to insure that the FPA got disabled.
0031  8439 ;                  d.  Halt.
0031  8440 ;          19.  Check the Trap register to insure that the FPA got disabled.
0031  8441 ;          20.  If the FPA is not disabled, then signal an error.
0031  8442 ;          21.  Set-up flag to expect a micro trap.
0031  8443 ;          22.  Initialize loop I for 256 opcodes (00-FF).
0031  8444 ;          23.  Store the index I in a location that will be used to load into the
0031  8445 ;               WDREG.  The value is used as the opcode.
0031  8446 ;          24.  Loop on error from here.
0031  8447 ;          25.  Execute the following test microcode:
0031  8448 ;                  a.  NOP to get the RDM up to speed.
0031  8449 ;                  b.  Read the WDREG into RTEMP2 (opcode).
0031  8450 ;                  c.  Load the operands into the LONLIT register.
0031  8451 ;                  d.  Store the operands into the Q register to prepare for ORing
0031  8452 ;                      the opcode with the operands.
0031  8453 ;                  e.  OR the opcode and operands together (XB0 data).
0031  8454 ;                  f.  Store XB0 data in M[TEMP8] in case needed for error callout.
0031  8455 ;                  g.  Zero the VA.
0031  8456 ;                  h.  Validate cache location 0 by first writing the data in
0031  8457 ;                      RTEMP2 then reading the data back.
0031  8458 ;                  i.  Update the VA by 4.
0031  8459 ;                  j.  Load the second operand data into the LONLIT register
0031  8460 ;                      (XB1 data).
0031  8461 ;                  k.  Store XB1 data in M[TEMP9] in case needed for error callout.
0031  8462 ;                  l.  Validate cache location 4 by first writing the data in
0031  8463 ;                      the LONLIT register then reading the data back.
0031  8464 ;                  m.  Load the PC with zero to cause a prefetch to take place.
0031  8465 ;                  n.  Do an IRD1 to init the IRD counter and prepare for instruc-
```

```
0031  8466 ;                    tion/operand specifier evaluation.
0031  8467 ;                 o.  Halt.
0031  8468 ;          26.  Perform the following test microcode.
0031  8469 ;                 a.  NOP to get the RDM up to speed.
0031  8470 ;                 b.  Do an IRDX and strobe the VBUS.
0031  8471 ;                 c.  Halt.
0031  8472 ;          27.  Compare the VBUS data with the expected value and bit position
0031  8473 ;               from the VBUS Table.
0031  8474 ;          28.  If the two do not equal, execute the following microcode and
0031  8475 ;               signal an error (this is done to set-up for printing RTEMP0 & 1
0031  8476 ;               on the error callout):
0031  8477 ;                 a.  NOP to get the RDM up to speed.
0031  8478 ;                 b.  Load RTEMP0 with the contents of M[TEMP8]
0031  8479 ;                 c.  Load RTEMP1 with the contents of M[TEMP9]
0031  8480 ;                 d.  Halt
0031  8481 ;          29.  Continue with loop I for the 256 VAX opcodes.
0031  8482 ;          30.  Go to Subtest 3.
0031  8483 ;
0031  8484 ;          Subtest 3
0031  8485 ;
0031  8486 ;          31.  Load the FPA enable bit (bit <5> equal to 1).
0031  8487 ;          32.  Loop on error from here.
0031  8488 ;          33.  Set-up to expect micro trap.
0031  8489 ;          34.  Execute the following microcode to enable the FPA:
0031  8490 ;                 a.  NOP to get the RDM up to speed.
0031  8491 ;                 b.  Read the RDM WDREG onto the W Bus and then write the W Bus
0031  8492 ;                     to the FPA.
0031  8493 ;                 c.  Perform a NO.FPA test to insure that the FPA got enabled.
0031  8494 ;                 d.  Halt.
0031  8495 ;          35.  Check the Trap register to insure that the FPA got enabled.
0031  8496 ;          36.  If the FPA is not enabled, then signal an error.
0031  8497 ;          37.  Set-up flag to expect a micro trap.
0031  8498 ;          38.  Initialize loop I for 256 opcodes (00-FF).
0031  8499 ;          39.  Store the index I in a location that will be used to load into the
0031  8500 ;               WDREG.  The value is used as the opcode.
0031  8501 ;          40.  Loop on error from here.
0031  8502 ;          41.  Set then Clear the DCLO bit in the RDCTRL register to perform an
0031  8503 ;               initialize.  This is done because when the test microcode
0031  8504 ;               executes, if the opcode was an FP opcode then there is a
0031  8505 ;               possibility that the processor could be setting there in an
0031  8506 ;               FPAWAIT state (M clock and D clock stalled).  If this were to
0031  8507 ;               happen then the cpu clocks would no longer run and the remaining
0031  8508 ;               test opcodes would fail.
0031  8509 ;          42.  Execute the following test microcode:
0031  8510 ;                 a.  NOP to get the RDM up to speed.
0031  8511 ;                 b.  Read the WDREG into RTEMP2 (opcode).
0031  8512 ;                 c.  Load the operands into the LONLIT register.
0031  8513 ;                 d.  Store the operands into the Q register to prepare for ORing
0031  8514 ;                     the opcode with the operands.
0031  8515 ;                 e.  OR the opcode and operands together (XB0 data).
0031  8516 ;                 f.  Store XB0 data in M[TEMP8] in case needed for error callout.
0031  8517 ;                 g.  Zero the VA.
0031  8518 ;                 h.  Validate cache location 0 by first writing the data in
0031  8519 ;                     RTEMP2 then reading the data back.
0031  8520 ;                 i.  Update the VA by 4.
```

```
0031  8521 ;                  j.  Load the second operand data into the LONLIT register
0031  8522 ;                      (XB1 data).
0031  8523 ;                  k.  Store XB1 data in M[TEMP9] in case needed for error callout.
0031  8524 ;                  l.  Validate cache location 4 by first writing the data in
0031  8525 ;                      the LONLIT register then reading the data back.
0031  8526 ;                  m.  Load the PC with zero to cause a prefetch to take place.
0031  8527 ;                  n.  Do an IRD1 to init the IRD counter and prepare for instruc-
0031  8528 ;                      tion/operand specifier evaluation.
0031  8529 ;                  o.  Halt.
0031  8530 ;          43.  Perform the following test microcode.
0031  8531 ;                  a.  NOP to get the RDM up to speed.
0031  8532 ;                  b.  Do an IRDX and strobe the VBUS.
0031  8533 ;                  c.  Halt.
0031  8534 ;          44.  Compare the VBUS data with the expected value and bit position
0031  8535 ;               from the VBUS Table.
0031  8536 ;          45.  If the two do not equal, execute the following microcode and
0031  8537 ;               signal an error (this is done to set-up for printing RTEMP0 & 1
0031  8538 ;               on the error callout):
0031  8539 ;                  a.  NOP to get the RDM up to speed.
0031  8540 ;                  b.  Load RTEMP0 with the contents of M[TEMP8]
0031  8541 ;                  c.  Load RTEMP1 with the contents of M[TEMP9]
0031  8542 ;                  d.  Halt
0031  8543 ;          46.  Continue with loop I for the 256 VAX opcodes.
0031  8544 ;          47.  Re-enable CTRL/P interrupts.
0031  8545 ;          48.  Skip to next tests.
0031  8546 ;
0031  8547 ; TEST PATTERNS
0031  8548 ;
0031  8549 ;          For this test the opcodes are generated by a simple count pattern.  The
0031  8550 ;          opcodes are based on the 256 base machine instructions as specified in
0031  8551 ;          the VAX architecture handbook.  The generated opcodes are written to
0031  8552 ;          the CPU and combined (ORed) with three bytes of operand data to form
0031  8553 ;          a machine instruction.  The operands all specify Register mode using
0031  8554 ;          registers 0-6 and are loaded from the LONLIT register.
0031  8555 ;
0031  8556 ;          Also required are two tables of expected VBUS values for the first IRDX
0031  8557 ;          cycle.  One table contains the VBUS values for the first IRDX when the
0031  8558 ;          FPA is disabled, the other for when it is enabled.
0031  8559 ;
0031  8560 ; LOGIC DESCRIPTION
0031  8561 ;
0031  8562 ;          This test will check the validity of the IRDX roms on the DPM module.
0031  8563 ;          The IRDX roms are addressed by bits <7:0> of the Instruction Register.
0031  8564 ;          Along with being addressed by the IR, there are three other signals
0031  8565 ;          that affect the address.  They are IRD CTR 0 H, FPA ENABLED L, and
0031  8566 ;          REG MODE H.  This test will examine every 8th(hex) address starting at
0031  8567 ;          address 3 with FPA disabled and starting at address 1 with FPA enabled.
0031  8568 ;
0031  8569 ; ASSUMPTIONS
0031  8570 ;
0031  8571 ;          This test assumes the following logic has already been tested; the
0031  8572 ;          CPU's data path, the cache, the XB, and the IR and OSR.
0031  8573 ;
0031  8574 ; ERROR DESCRIPTION
0031  8575 ;
```

```
                        0031  8576 ;       Subtest 1
                        0031  8577 ;
                        0031  8578 ;               Expected CS Address
                        0031  8579 ;               Received CS Address
                        0031  8580 ;
                        0031  8581 ;       Subtest 2
                        0031  8582 ;
                        0031  8583 ;               First IFERROR
                        0031  8584 ;
                        0031  8585 ;                       Expected CS Address
                        0031  8586 ;                       Received CS Address
                        0031  8587 ;
                        0031  8588 ;               Second IFERROR
                        0031  8589 ;
                        0031  8590 ;                       Expected VBUS ( 00000ynn )      y  = VBUS bit value
                        0031  8591 ;                       Received VBUS ( 00000ynn )      nn = VBUS bit number
                        0031  8592 ;                       Loop I count
                        0031  8593 ;                       Contents of XB0
                        0031  8594 ;                       Contents of XB1
                        0031  8595 ;
                        0031  8596 ;       Subtest 3
                        0031  8597 ;
                        0031  8598 ;               First IFERROR
                        0031  8599 ;
                        0031  8600 ;                       Expected CS Address
                        0031  8601 ;                       Received CS Address
                        0031  8602 ;
                        0031  8603 ;               Second IFERROR
                        0031  8604 ;
                        0031  8605 ;                       Expected VBUS ( 00000ynn )      y  = VBUS bit value
                        0031  8606 ;                       Received VBUS ( 00000ynn )      nn = VBUS bit number
                        0031  8607 ;                       Loop I count
                        0031  8608 ;                       Contents of XB0
                        0031  8609 ;                       Contents of XB1
                        0031  8610 ;--
                        0031  8611 ;*******************************************************************************
                        0031  8612
                        0031  8613          INITIALIZE                              ; initialize the comet cpu
                        0033  8614          FETCH           <^X19>                  ; disable ctrl-p interrupts
                        0038  8615          BRSTCLK         <^X1E>                  ; ...
                        003C  8616          FETCH           <^X16>                  ; execute a but/no.fpa to check
                        0041  8617          BRSTCLK         <^X18>                  ; for fpa present
                        0045  8618          CMPREGMSK       CSTRP,1$,,2$,,W         ; is the fpa present?
                        0051  8619          SKIP            100$,ONERROR            ; if so then skip to subtest 2
                        0058  8620
                        0058  8621          SUBTEST         1
                        005B
                              ;////////////////////////////////////////////////////////////////////////////////
                        005B  8622 ;+
                        005B  8623 ;
                        005B  8624 ; FPA is not present.
                        005B  8625 ;
                        005B  8626 ;-
                        005B  8627          EXPUTRAP                                ; else expect non-dcs addresses
                        005D  8628          LOOP            I,1,256                 ; loop for 256 opcodes
```

```
                0064  8629          SAVEINDEX       I,9$                    ; save the loop count
                0069  8630          LOADREG         WDREG,9$,,L             ; load wdreg with loop count
                0071  8631          ERRLOOP                                 ; Loop on error from here
                0073  8632          FETCH           0                       ; load the istrm data and do an
                0078  8633          BRSTCLK         <^X10>,INH              ; ird1TST
                007C  8634          FETCH           <^X11>                  ; execute the irdx
                0081  8635          BRSTCLK         <^X13>,INH              ; ...
                0085  8636          CMPVBUS         30$,I                   ; check VBUS bit for an error
                008A  8637          SKIP            50$,NOERROR             ; continue testing if no error
                0091  8638          FETCH           <^X25>                  ; start dcs program
                0096  8639          BRSTCLK         <^X28>                  ; end dcs program
                009A  8640          IFERROR         2,<<IRD>>,<<MIC>>       ; if bad address, signal error
                00A1  8641 50$:     ENDLOOP         I                       ; continue loop
                00A4  8642          FETCH           <^X1F>                  ; re-enable ctrl-p interrupts
                00A9  8643          BRSTCLK         <^X24>                  ; ...
                00AD  8644          SKIP                                    ; skip on to the next test
                00B4  8645
                00B4  8646 100$:    SUBTEST         2
                00B7
                00B7          ;////////////////////////////////////////////////////////////////////////////////
                00B7  8647 ;*
                00B7  8648 ;
                00B7  8649 ; FPA present but not enabled
                00B7  8650 ;
                00B7  8651 ;-
                00B7  8652          LOADREG         WDREG,7$,,L             ; load fpa disable bit
                00BF  8653          ERRLOOP                                 ; loop on error here
                00C1  8654          EXPUTRAP                                ; expect micro trap
                00C3  8655          FETCH           0                       ; start dcs program
                00C8  8656          BRSTCLK         <^X10>,INH              ; end dcs program
                00CC  8657          FETCH           <^X14>                  ; execute the disable
                00D1  8658          BRSTCLK         <^X18>,INH              ; and read back fpa status
                00D5  8659          CMPREGMSK       CSTRP,3$,,5$,,W         ; is the fpa disabled?
                00E1  8660          IFERROR         ,,<FPA>                 ; if not, signal an error
                00E7  8661          EXPUTRAP                                ; else expect non-dcs addresses
                00E9  8662          LOOP            I,1,256                 ; loop for 256 opcodes
                00F0  8663          SAVEINDEX       I,9$                    ; save the loop count
                00F5  8664          LOADREG         WDREG,9$,,L             ; load wdreg with loop count
                00FD  8665          ERRLOOP                                 ; Loop on error from here
                00FF  8666          FETCH           0                       ; execute a but/ird1tst
                0104  8667          BRSTCLK         <^X10>,INH              ; ...
                0108  8668          FETCH           <^X11>                  ; execute an irdx
                010D  8669          BRSTCLK         <^X13>,INH              ; ...
                0111  8670          CMPVBUS         30$,I                   ; check VBUS bit for an error
                0116  8671          SKIP            51$,NOERROR             ; continue testing if no error
                011D  8672          FETCH           <^X25>                  ; start dcs program
                0122  8673          BRSTCLK         <^X28>                  ; end dcs program
                0126  8674          IFERROR         2,<<IRD>>,<<MIC>>       ; if bad address, signal error
                012D  8675 51$:     ENDLOOP         I                       ; continue loop
                0130  8676
                0130  8677          SUBTEST         3
                0133
                0133          ;////////////////////////////////////////////////////////////////////////////////
                0133  8678 ;*
                0133  8679 ;
```

```
                        0133  8680 ; FPA present and enabled
                        0133  8681 ;
                        0133  8682 ;-
                        0133  8683          LOADREG         WDREG,8$,,L                     ; load fpa enable bit
                        013B  8684          ERRLOOP                                         ; loop on error from here
                        013D  8685          FETCH           <^X14>                          ; enable the fpa
                        0142  8686          BRSTCLK         <^X18>,INH                      ; and read back fpa enable
                        0146  8687          CMPREGMSK       CSTRP,4$,,5$,,W                 ; is the fpa enabled?
                        0152  8688          IFERROR         ,,<FPA>                         ; if not, signal an error
                        0158  8689          EXPUTRAP                                        ; else expect non-dcs addresses
                        015A  8690          LOOP            I,1,256                         ; loop for 256 opcodes
                        0161  8691          SAVEINDEX       I,9$                            ; save the loop count
                        0166  8692          LOADREG         WDREG,9$,,L                     ; load wdreg with loop count
                        016E  8693          ERRLOOP                                         ; Loop on error from here
                        0170  8694          LOADREG         RDCTRL,10$,,B                   ; assert dclo
                        0178  8695          LOADREG         RDCTRL,11$,,B                   ; de-assert dclo
                        0180  8696          FETCH           0                               ; execute a but/ird1tst
                        0185  8697          BRSTCLK         <^X10>,INH                      ; ...
                        0189  8698          FETCH           <^X11>                          ; execute the but/irdx
                        018E  8699          BRSTCLK         <^X13>,INH                      ; ...
                        0192  8700          CMPVBUS         31$,I                           ; check VBUS bit for an error
                        0197  8701          SKIP            52$,NOERROR                     ; continue testing if no error
                        019E  8702          FETCH           <^X25>                          ; start dcs program
                        01A3  8703          BRSTCLK         <^X28>                          ; end dcs program
                        01A7  8704          IFERROR         2,<<IRD>>,<<MIC>>               ; if bad address, signal error
                        01AE  8705 52$:     ENDLOOP         I                               ; continue loop
                        01B1  8706          FETCH           <^X1F>                          ; re-enable ctrl-p interrupts
                        01B6  8707          BRSTCLK         <^X24>                          ; ...
                        01BA  8708          SKIP                                            ; end of test
                        01C1  8709
                        01C1  8710 BEGIN_TEST_DATA
                        01C1
                        01C1          ;-----------------------------------------------------------------------------
                        01C1  8711
                1802    01C1  8712 1$:      .WORD   ^X1802
                3FFE    01C3  8713 2$:      .WORD   ^X3FFE
                        01C5  8714
                1801    01C5  8715 3$:      .WORD   ^X1801
                1800    01C7  8716 4$:      .WORD   ^X1800
                3FFD    01C9  8717 5$:      .WORD   ^X3FFD
                        01CB  8718
                3FFF    01CB  8719 6$:      .WORD   ^X3FFF
                        01CD  8720
            00000000    01CD  8721 7$:      .LONG   0
            00000020    01D1  8722 8$:      .LONG   ^X00000020
                        01D5  8723
            00000000    01D5  8724 9$:      .LONG   0
                  40    01D9  8725 10$:     .BYTE   DC_LOW
                  00    01DA  8726 11$:     .BYTE   ^X00
                        01DB  8727
                        01DB  8728
                        01DB  8729 ;
                        01DB  8730 ; Expected VBUS data for Subtests 1 and 2
                        01DB  8731 ;
                        01DB  8732 ; The following is a table of all the expected VBUS values for the first IRDX.
```

```
                    01DB  8733 ; A 1 means the ROM OS INH H signal is asserted and a 0 means it is not
                    01DB  8734 ; asserted.
                    01DB  8735 ;
                    01DB  8736
 01                 01DB  8737 30$:     .BYTE   ^X01
                    01DC  8738          VBUSDATA        <^X1E>,1                ;00
 01                 01DD  8739          .BYTE   ^X01
                    01DE  8740          VBUSDATA        <^X1E>,1                ;01
 01                 01DF  8741          .BYTE   ^X01
                    01E0  8742          VBUSDATA        <^X1E>,1                ;02
 01                 01E1  8743          .BYTE   ^X01
                    01E2  8744          VBUSDATA        <^X1E>,1                ;03
 01                 01E3  8745          .BYTE   ^X01
                    01E4  8746          VBUSDATA        <^X1E>,1                ;04
 01                 01E5  8747          .BYTE   ^X01
                    01E6  8748          VBUSDATA        <^X1E>,1                ;05
 01                 01E7  8749          .BYTE   ^X01
                    01E8  8750          VBUSDATA        <^X1E>,1                ;06
 01                 01E9  8751          .BYTE   ^X01
                    01EA  8752          VBUSDATA        <^X1E>,1                ;07
 01                 01EB  8753          .BYTE   ^X01
                    01EC  8754          VBUSDATA        <^X1E>,1                ;08
 01                 01ED  8755          .BYTE   ^X01
                    01EE  8756          VBUSDATA        <^X1E>,1                ;09
 01                 01EF  8757          .BYTE   ^X01
                    01F0  8758          VBUSDATA        <^X1E>,0                ;0A
 01                 01F1  8759          .BYTE   ^X01
                    01F2  8760          VBUSDATA        <^X1E>,1                ;0B
 01                 01F3  8761          .BYTE   ^X01
                    01F4  8762          VBUSDATA        <^X1E>,0                ;0C
 01                 01F5  8763          .BYTE   ^X01
                    01F6  8764          VBUSDATA        <^X1E>,0                ;0D
 01                 01F7  8765          .BYTE   ^X01
                    01F8  8766          VBUSDATA        <^X1E>,0                ;0E
 01                 01F9  8767          .BYTE   ^X01
                    01FA  8768          VBUSDATA        <^X1E>,0                ;0F
 01                 01FB  8769          .BYTE   ^X01
                    01FC  8770          VBUSDATA        <^X1E>,1                ;10
 01                 01FD  8771          .BYTE   ^X01
                    01FE  8772          VBUSDATA        <^X1E>,1                ;11
 01                 01FF  8773          .BYTE   ^X01
                    0200  8774          VBUSDATA        <^X1E>,1                ;12
 01                 0201  8775          .BYTE   ^X01
                    0202  8776          VBUSDATA        <^X1E>,1                ;13
 01                 0203  8777          .BYTE   ^X01
                    0204  8778          VBUSDATA        <^X1E>,1                ;14
 01                 0205  8779          .BYTE   ^X01
                    0206  8780          VBUSDATA        <^X1E>,1                ;15
 01                 0207  8781          .BYTE   ^X01
                    0208  8782          VBUSDATA        <^X1E>,1                ;16
 01                 0209  8783          .BYTE   ^X01
                    020A  8784          VBUSDATA        <^X1E>,1                ;17
 01                 020B  8785          .BYTE   ^X01
                    020C  8786          VBUSDATA        <^X1E>,1                ;18
 01                 020D  8787          .BYTE   ^X01
```

```
                020E  8788          VBUSDATA        <^X1E>,1                ;19
01              020F  8789          .BYTE   ^X01
                0210  8790          VBUSDATA        <^X1E>,1                ;1A
01              0211  8791          .BYTE   ^X01
                0212  8792          VBUSDATA        <^X1E>,1                ;1B
01              0213  8793          .BYTE   ^X01
                0214  8794          VBUSDATA        <^X1E>,1                ;1C
01              0215  8795          .BYTE   ^X01
                0216  8796          VBUSDATA        <^X1E>,1                ;1D
01              0217  8797          .BYTE   ^X01
                0218  8798          VBUSDATA        <^X1E>,1                ;1E
01              0219  8799          .BYTE   ^X01
                021A  8800          VBUSDATA        <^X1E>,1                ;1F
01              021B  8801          .BYTE   ^X01
                021C  8802          VBUSDATA        <^X1E>,1                ;20
01              021D  8803          .BYTE   ^X01
                021E  8804          VBUSDATA        <^X1E>,1                ;21
01              021F  8805          .BYTE   ^X01
                0220  8806          VBUSDATA        <^X1E>,1                ;22
01              0221  8807          .BYTE   ^X01
                0222  8808          VBUSDATA        <^X1E>,1                ;23
01              0223  8809          .BYTE   ^X01
                0224  8810          VBUSDATA        <^X1E>,1                ;24
01              0225  8811          .BYTE   ^X01
                0226  8812          VBUSDATA        <^X1E>,1                ;25
01              0227  8813          .BYTE   ^X01
                0228  8814          VBUSDATA        <^X1E>,1                ;26
01              0229  8815          .BYTE   ^X01
                022A  8816          VBUSDATA        <^X1E>,1                ;27
01              022B  8817          .BYTE   ^X01
                022C  8818          VBUSDATA        <^X1E>,1                ;28
01              022D  8819          .BYTE   ^X01
                022E  8820          VBUSDATA        <^X1E>,1                ;29
01              022F  8821          .BYTE   ^X01
                0230  8822          VBUSDATA        <^X1E>,1                ;2A
01              0231  8823          .BYTE   ^X01
                0232  8824          VBUSDATA        <^X1E>,1                ;2B
01              0233  8825          .BYTE   ^X01
                0234  8826          VBUSDATA        <^X1E>,1                ;2C
01              0235  8827          .BYTE   ^X01
                0236  8828          VBUSDATA        <^X1E>,1                ;2D
01              0237  8829          .BYTE   ^X01
                0238  8830          VBUSDATA        <^X1E>,1                ;2E
01              0239  8831          .BYTE   ^X01
                023A  8832          VBUSDATA        <^X1E>,1                ;2F
01              023B  8833          .BYTE   ^X01
                023C  8834          VBUSDATA        <^X1E>,1                ;30
01              023D  8835          .BYTE   ^X01
                023E  8836          VBUSDATA        <^X1E>,1                ;31
01              023F  8837          .BYTE   ^X01
                0240  8838          VBUSDATA        <^X1E>,1                ;32
01              0241  8839          .BYTE   ^X01
                0242  8840          VBUSDATA        <^X1E>,0                ;33
01              0243  8841          .BYTE   ^X01
                0244  8842          VBUSDATA        <^X1E>,1                ;34
```

```
                                01  0245  8843          .BYTE   ^X01
                                    0246  8844          VBUSDATA        <^X1E>,1                ;35
                                01  0247  8845          .BYTE   ^X01
                                    0248  8846          VBUSDATA        <^X1E>,1                ;36
                                01  0249  8847          .BYTE   ^X01
                                    024A  8848          VBUSDATA        <^X1E>,1                ;37
                                01  024B  8849          .BYTE   ^X01
                                    024C  8850          VBUSDATA        <^X1E>,1                ;38
                                01  024D  8851          .BYTE   ^X01
                                    024E  8852          VBUSDATA        <^X1E>,1                ;39
                                01  024F  8853          .BYTE   ^X01
                                    0250  8854          VBUSDATA        <^X1E>,1                ;3A
                                01  0251  8855          .BYTE   ^X01
                                    0252  8856          VBUSDATA        <^X1E>,1                ;3B
                                01  0253  8857          .BYTE   ^X01
                                    0254  8858          VBUSDATA        <^X1E>,1                ;3C
                                01  0255  8859          .BYTE   ^X01
                                    0256  8860          VBUSDATA        <^X1E>,0                ;3D
                                01  0257  8861          .BYTE   ^X01
                                    0258  8862          VBUSDATA        <^X1E>,0                ;3E
                                01  0259  8863          .BYTE   ^X01
                                    025A  8864          VBUSDATA        <^X1E>,1                ;3F
                                01  025B  8865          .BYTE   ^X01
                                    025C  8866          VBUSDATA        <^X1E>,0                ;40
                                01  025D  8867          .BYTE   ^X01
                                    025E  8868          VBUSDATA        <^X1E>,0                ;41
                                01  025F  8869          .BYTE   ^X01
                                    0260  8870          VBUSDATA        <^X1E>,1                ;42
                                01  0261  8871          .BYTE   ^X01
                                    0262  8872          VBUSDATA        <^X1E>,1                ;43
                                01  0263  8873          .BYTE   ^X01
                                    0264  8874          VBUSDATA        <^X1E>,0                ;44
                                01  0265  8875          .BYTE   ^X01
                                    0266  8876          VBUSDATA        <^X1E>,0                ;45
                                01  0267  8877          .BYTE   ^X01
                                    0268  8878          VBUSDATA        <^X1E>,0                ;46
                                01  0269  8879          .BYTE   ^X01
                                    026A  8880          VBUSDATA        <^X1E>,0                ;47
                                01  026B  8881          .BYTE   ^X01
                                    026C  8882          VBUSDATA        <^X1E>,0                ;48
                                01  026D  8883          .BYTE   ^X01
                                    026E  8884          VBUSDATA        <^X1E>,0                ;49
                                01  026F  8885          .BYTE   ^X01
                                    0270  8886          VBUSDATA        <^X1E>,0                ;4A
                                01  0271  8887          .BYTE   ^X01
                                    0272  8888          VBUSDATA        <^X1E>,0                ;4B
                                01  0273  8889          .BYTE   ^X01
                                    0274  8890          VBUSDATA        <^X1E>,1                ;4C
                                01  0275  8891          .BYTE   ^X01
                                    0276  8892          VBUSDATA        <^X1E>,1                ;4D
                                01  0277  8893          .BYTE   ^X01
                                    0278  8894          VBUSDATA        <^X1E>,0                ;4E
                                01  0279  8895          .BYTE   ^X01
                                    027A  8896          VBUSDATA        <^X1E>,0                ;4F
                                01  027B  8897          .BYTE   ^X01
```

```
                    027C  8898          VBUSDATA        <^X1E>,0                ;50
                01  027D  8899          .BYTE   ^X01
                    027E  8900          VBUSDATA        <^X1E>,0                ;51
                01  027F  8901          .BYTE   ^X01
                    0280  8902          VBUSDATA        <^X1E>,0                ;52
                01  0281  8903          .BYTE   ^X01
                    0282  8904          VBUSDATA        <^X1E>,1                ;53
                01  0283  8905          .BYTE   ^X01
                    0284  8906          VBUSDATA        <^X1E>,0                ;54
                01  0285  8907          .BYTE   ^X01
                    0286  8908          VBUSDATA        <^X1E>,0                ;55
                01  0287  8909          .BYTE   ^X01
                    0288  8910          VBUSDATA        <^X1E>,0                ;56
                01  0289  8911          .BYTE   ^X01
                    028A  8912          VBUSDATA        <^X1E>,1                ;57
                01  028B  8913          .BYTE   ^X01
                    028C  8914          VBUSDATA        <^X1E>,0                ;58
                01  028D  8915          .BYTE   ^X01
                    028E  8916          VBUSDATA        <^X1E>,1                ;59
                01  028F  8917          .BYTE   ^X01
                    0290  8918          VBUSDATA        <^X1E>,1                ;5A
                01  0291  8919          .BYTE   ^X01
                    0292  8920          VBUSDATA        <^X1E>,1                ;5B
                01  0293  8921          .BYTE   ^X01
                    0294  8922          VBUSDATA        <^X1E>,0                ;5C
                01  0295  8923          .BYTE   ^X01
                    0296  8924          VBUSDATA        <^X1E>,0                ;5D
                01  0297  8925          .BYTE   ^X01
                    0298  8926          VBUSDATA        <^X1E>,0                ;5E
                01  0299  8927          .BYTE   ^X01
                    029A  8928          VBUSDATA        <^X1E>,0                ;5F
                01  029B  8929          .BYTE   ^X01
                    029C  8930          VBUSDATA        <^X1E>,1                ;60
                01  029D  8931          .BYTE   ^X01
                    029E  8932          VBUSDATA        <^X1E>,1                ;61
                01  029F  8933          .BYTE   ^X01
                    02A0  8934          VBUSDATA        <^X1E>,1                ;62
                01  02A1  8935          .BYTE   ^X01
                    02A2  8936          VBUSDATA        <^X1E>,1                ;63
                01  02A3  8937          .BYTE   ^X01
                    02A4  8938          VBUSDATA        <^X1E>,1                ;64
                01  02A5  8939          .BYTE   ^X01
                    02A6  8940          VBUSDATA        <^X1E>,1                ;65
                01  02A7  8941          .BYTE   ^X01
                    02A8  8942          VBUSDATA        <^X1E>,1                ;66
                01  02A9  8943          .BYTE   ^X01
                    02AA  8944          VBUSDATA        <^X1E>,1                ;67
                01  02AB  8945          .BYTE   ^X01
                    02AC  8946          VBUSDATA        <^X1E>,1                ;68
                01  02AD  8947          .BYTE   ^X01
                    02AE  8948          VBUSDATA        <^X1E>,1                ;69
                01  02AF  8949          .BYTE   ^X01
                    02B0  8950          VBUSDATA        <^X1E>,1                ;6A
                01  02B1  8951          .BYTE   ^X01
                    02B2  8952          VBUSDATA        <^X1E>,1                ;6B
```

```
01  02B3  8953          .BYTE   ^X01
    02B4  8954          VBUSDATA        <^X1E>,1                ;6C
01  02B5  8955          .BYTE   ^X01
    02B6  8956          VBUSDATA        <^X1E>,1                ;6D
01  02B7  8957          .BYTE   ^X01
    02B8  8958          VBUSDATA        <^X1E>,1                ;6E
01  02B9  8959          .BYTE   ^X01
    02BA  8960          VBUSDATA        <^X1E>,1                ;6F
01  02BB  8961          .BYTE   ^X01
    02BC  8962          VBUSDATA        <^X1E>,0                ;70
01  02BD  8963          .BYTE   ^X01
    02BE  8964          VBUSDATA        <^X1E>,1                ;71
01  02BF  8965          .BYTE   ^X01
    02C0  8966          VBUSDATA        <^X1E>,0                ;72
01  02C1  8967          .BYTE   ^X01
    02C2  8968          VBUSDATA        <^X1E>,1                ;73
01  02C3  8969          .BYTE   ^X01
    02C4  8970          VBUSDATA        <^X1E>,1                ;74
01  02C5  8971          .BYTE   ^X01
    02C6  8972          VBUSDATA        <^X1E>,1                ;75
01  02C7  8973          .BYTE   ^X01
    02C8  8974          VBUSDATA        <^X1E>,0                ;76
01  02C9  8975          .BYTE   ^X01
    02CA  8976          VBUSDATA        <^X1E>,1                ;77
01  02CB  8977          .BYTE   ^X01
    02CC  8978          VBUSDATA        <^X1E>,0                ;78
01  02CD  8979          .BYTE   ^X01
    02CE  8980          VBUSDATA        <^X1E>,0                ;79
01  02CF  8981          .BYTE   ^X01
    02D0  8982          VBUSDATA        <^X1E>,0                ;7A
01  02D1  8983          .BYTE   ^X01
    02D2  8984          VBUSDATA        <^X1E>,0                ;7B
01  02D3  8985          .BYTE   ^X01
    02D4  8986          VBUSDATA        <^X1E>,1                ;7C
01  02D5  8987          .BYTE   ^X01
    02D6  8988          VBUSDATA        <^X1E>,0                ;7D
01  02D7  8989          .BYTE   ^X01
    02D8  8990          VBUSDATA        <^X1E>,0                ;7E
01  02D9  8991          .BYTE   ^X01
    02DA  8992          VBUSDATA        <^X1E>,1                ;7F
01  02DB  8993          .BYTE   ^X01
    02DC  8994          VBUSDATA        <^X1E>,0                ;80
01  02DD  8995          .BYTE   ^X01
    02DE  8996          VBUSDATA        <^X1E>,0                ;81
01  02DF  8997          .BYTE   ^X01
    02E0  8998          VBUSDATA        <^X1E>,0                ;82
01  02E1  8999          .BYTE   ^X01
    02E2  9000          VBUSDATA        <^X1E>,0                ;83
01  02E3  9001          .BYTE   ^X01
    02E4  9002          VBUSDATA        <^X1E>,0                ;84
01  02E5  9003          .BYTE   ^X01
    02E6  9004          VBUSDATA        <^X1E>,0                ;85
01  02E7  9005          .BYTE   ^X01
    02E8  9006          VBUSDATA        <^X1E>,1                ;86
01  02E9  9007          .BYTE   ^X01
```

```
                02EA  9008          VBUSDATA        <^X1E>,1                ;87
 01             02EB  9009          .BYTE   ^X01
                02EC  9010          VBUSDATA        <^X1E>,0                ;88
 01             02ED  9011          .BYTE   ^X01
                02EE  9012          VBUSDATA        <^X1E>,0                ;89
 01             02EF  9013          .BYTE   ^X01
                02F0  9014          VBUSDATA        <^X1E>,0                ;8A
 01             02F1  9015          .BYTE   ^X01
                02F2  9016          VBUSDATA        <^X1E>,0                ;8B
 01             02F3  9017          .BYTE   ^X01
                02F4  9018          VBUSDATA        <^X1E>,0                ;8C
 01             02F5  9019          .BYTE   ^X01
                02F6  9020          VBUSDATA        <^X1E>,0                ;8D
 01             02F7  9021          .BYTE   ^X01
                02F8  9022          VBUSDATA        <^X1E>,0                ;8E
 01             02F9  9023          .BYTE   ^X01
                02FA  9024          VBUSDATA        <^X1E>,0                ;8F
 01             02FB  9025          .BYTE   ^X01
                02FC  9026          VBUSDATA        <^X1E>,0                ;90
 01             02FD  9027          .BYTE   ^X01
                02FE  9028          VBUSDATA        <^X1E>,0                ;91
 01             02FF  9029          .BYTE   ^X01
                0300  9030          VBUSDATA        <^X1E>,0                ;92
 01             0301  9031          .BYTE   ^X01
                0302  9032          VBUSDATA        <^X1E>,0                ;93
 01             0303  9033          .BYTE   ^X01
                0304  9034          VBUSDATA        <^X1E>,1                ;94
 01             0305  9035          .BYTE   ^X01
                0306  9036          VBUSDATA        <^X1E>,1                ;95
 01             0307  9037          .BYTE   ^X01
                0308  9038          VBUSDATA        <^X1E>,1                ;96
 01             0309  9039          .BYTE   ^X01
                030A  9040          VBUSDATA        <^X1E>,1                ;97
 01             030B  9041          .BYTE   ^X01
                030C  9042          VBUSDATA        <^X1E>,1                ;98
 01             030D  9043          .BYTE   ^X01
                030E  9044          VBUSDATA        <^X1E>,1                ;99
 01             030F  9045          .BYTE   ^X01
                0310  9046          VBUSDATA        <^X1E>,1                ;9A
 01             0311  9047          .BYTE   ^X01
                0312  9048          VBUSDATA        <^X1E>,1                ;9B
 01             0313  9049          .BYTE   ^X01
                0314  9050          VBUSDATA        <^X1E>,0                ;9C
 01             0315  9051          .BYTE   ^X01
                0316  9052          VBUSDATA        <^X1E>,0                ;9D
 01             0317  9053          .BYTE   ^X01
                0318  9054          VBUSDATA        <^X1E>,0                ;9E
 01             0319  9055          .BYTE   ^X01
                031A  9056          VBUSDATA        <^X1E>,1                ;9F
 01             031B  9057          .BYTE   ^X01
                031C  9058          VBUSDATA        <^X1E>,0                ;A0
 01             031D  9059          .BYTE   ^X01
                031E  9060          VBUSDATA        <^X1E>,0                ;A1
 01             031F  9061          .BYTE   ^X01
                0320  9062          VBUSDATA        <^X1E>,0                ;A2
```

```
01  0321  9063          .BYTE   ^X01
    0322  9064          VBUSDATA        <^X1E>,0                ;A3
01  0323  9065          .BYTE   ^X01
    0324  9066          VBUSDATA        <^X1E>,0                ;A4
01  0325  9067          .BYTE   ^X01
    0326  9068          VBUSDATA        <^X1E>,0                ;A5
01  0327  9069          .BYTE   ^X01
    0328  9070          VBUSDATA        <^X1E>,1                ;A6
01  0329  9071          .BYTE   ^X01
    032A  9072          VBUSDATA        <^X1E>,1                ;A7
01  032B  9073          .BYTE   ^X01
    032C  9074          VBUSDATA        <^X1E>,0                ;A8
01  032D  9075          .BYTE   ^X01
    032E  9076          VBUSDATA        <^X1E>,0                ;A9
01  032F  9077          .BYTE   ^X01
    0330  9078          VBUSDATA        <^X1E>,0                ;AA
01  0331  9079          .BYTE   ^X01
    0332  9080          VBUSDATA        <^X1E>,0                ;AB
01  0333  9081          .BYTE   ^X01
    0334  9082          VBUSDATA        <^X1E>,0                ;AC
01  0335  9083          .BYTE   ^X01
    0336  9084          VBUSDATA        <^X1E>,0                ;AD
01  0337  9085          .BYTE   ^X01
    0338  9086          VBUSDATA        <^X1E>,0                ;AE
01  0339  9087          .BYTE   ^X01
    033A  9088          VBUSDATA        <^X1E>,0                ;AF
01  033B  9089          .BYTE   ^X01
    033C  9090          VBUSDATA        <^X1E>,0                ;B0
01  033D  9091          .BYTE   ^X01
    033E  9092          VBUSDATA        <^X1E>,0                ;B1
01  033F  9093          .BYTE   ^X01
    0340  9094          VBUSDATA        <^X1E>,0                ;B2
01  0341  9095          .BYTE   ^X01
    0342  9096          VBUSDATA        <^X1E>,0                ;B3
01  0343  9097          .BYTE   ^X01
    0344  9098          VBUSDATA        <^X1E>,1                ;B4
01  0345  9099          .BYTE   ^X01
    0346  9100          VBUSDATA        <^X1E>,1                ;B5
01  0347  9101          .BYTE   ^X01
    0348  9102          VBUSDATA        <^X1E>,1                ;B6
01  0349  9103          .BYTE   ^X01
    034A  9104          VBUSDATA        <^X1E>,1                ;B7
01  034B  9105          .BYTE   ^X01
    034C  9106          VBUSDATA        <^X1E>,1                ;B8
01  034D  9107          .BYTE   ^X01
    034E  9108          VBUSDATA        <^X1E>,1                ;B9
01  034F  9109          .BYTE   ^X01
    0350  9110          VBUSDATA        <^X1E>,1                ;BA
01  0351  9111          .BYTE   ^X01
    0352  9112          VBUSDATA        <^X1E>,1                ;BB
01  0353  9113          .BYTE   ^X01
    0354  9114          VBUSDATA        <^X1E>,1                ;BC
01  0355  9115          .BYTE   ^X01
    0356  9116          VBUSDATA        <^X1E>,1                ;BD
01  0357  9117          .BYTE   ^X01
```

```
                0358  9118          VBUSDATA        <^X1E>,1                ;BE
01              0359  9119          .BYTE   ^X01
                035A  9120          VBUSDATA        <^X1E>,1                ;BF
01              035B  9121          .BYTE   ^X01
                035C  9122          VBUSDATA        <^X1E>,0                ;C0
01              035D  9123          .BYTE   ^X01
                035E  9124          VBUSDATA        <^X1E>,0                ;C1
01              035F  9125          .BYTE   ^X01
                0360  9126          VBUSDATA        <^X1E>,0                ;C2
01              0361  9127          .BYTE   ^X01
                0362  9128          VBUSDATA        <^X1E>,0                ;C3
01              0363  9129          .BYTE   ^X01
                0364  9130          VBUSDATA        <^X1E>,0                ;C4
01              0365  9131          .BYTE   ^X01
                0366  9132          VBUSDATA        <^X1E>,0                ;C5
01              0367  9133          .BYTE   ^X01
                0368  9134          VBUSDATA        <^X1E>,1                ;C6
01              0369  9135          .BYTE   ^X01
                036A  9136          VBUSDATA        <^X1E>,1                ;C7
01              036B  9137          .BYTE   ^X01
                036C  9138          VBUSDATA        <^X1E>,0                ;C8
01              036D  9139          .BYTE   ^X01
                036E  9140          VBUSDATA        <^X1E>,0                ;C9
01              036F  9141          .BYTE   ^X01
                0370  9142          VBUSDATA        <^X1E>,0                ;CA
01              0371  9143          .BYTE   ^X01
                0372  9144          VBUSDATA        <^X1E>,0                ;CB
01              0373  9145          .BYTE   ^X01
                0374  9146          VBUSDATA        <^X1E>,0                ;CC
01              0375  9147          .BYTE   ^X01
                0376  9148          VBUSDATA        <^X1E>,0                ;CD
01              0377  9149          .BYTE   ^X01
                0378  9150          VBUSDATA        <^X1E>,0                ;CE
01              0379  9151          .BYTE   ^X01
                037A  9152          VBUSDATA        <^X1E>,0                ;CF
01              037B  9153          .BYTE   ^X01
                037C  9154          VBUSDATA        <^X1E>,0                ;D0
01              037D  9155          .BYTE   ^X01
                037E  9156          VBUSDATA        <^X1E>,0                ;D1
01              037F  9157          .BYTE   ^X01
                0380  9158          VBUSDATA        <^X1E>,0                ;D2
01              0381  9159          .BYTE   ^X01
                0382  9160          VBUSDATA        <^X1E>,0                ;D3
01              0383  9161          .BYTE   ^X01
                0384  9162          VBUSDATA        <^X1E>,1                ;D4
01              0385  9163          .BYTE   ^X01
                0386  9164          VBUSDATA        <^X1E>,1                ;D5
01              0387  9165          .BYTE   ^X01
                0388  9166          VBUSDATA        <^X1E>,1                ;D6
01              0389  9167          .BYTE   ^X01
                038A  9168          VBUSDATA        <^X1E>,1                ;D7
01              038B  9169          .BYTE   ^X01
                038C  9170          VBUSDATA        <^X1E>,0                ;D8
01              038D  9171          .BYTE   ^X01
                038E  9172          VBUSDATA        <^X1E>,0                ;D9
```

```
01  038F  9173          .BYTE   ^X01
    0390  9174          VBUSDATA        <^X1E>,0                ;DA
01  0391  9175          .BYTE   ^X01
    0392  9176          VBUSDATA        <^X1E>,1                ;DB
01  0393  9177          .BYTE   ^X01
    0394  9178          VBUSDATA        <^X1E>,1                ;DC
01  0395  9179          .BYTE   ^X01
    0396  9180          VBUSDATA        <^X1E>,1                ;DD
01  0397  9181          .BYTE   ^X01
    0398  9182          VBUSDATA        <^X1E>,0                ;DE
01  0399  9183          .BYTE   ^X01
    039A  9184          VBUSDATA        <^X1E>,1                ;DF
01  039B  9185          .BYTE   ^X01
    039C  9186          VBUSDATA        <^X1E>,0                ;E0
01  039D  9187          .BYTE   ^X01
    039E  9188          VBUSDATA        <^X1E>,0                ;E1
01  039F  9189          .BYTE   ^X01
    03A0  9190          VBUSDATA        <^X1E>,0                ;E2
01  03A1  9191          .BYTE   ^X01
    03A2  9192          VBUSDATA        <^X1E>,0                ;E3
01  03A3  9193          .BYTE   ^X01
    03A4  9194          VBUSDATA        <^X1E>,0                ;E4
01  03A5  9195          .BYTE   ^X01
    03A6  9196          VBUSDATA        <^X1E>,0                ;E5
01  03A7  9197          .BYTE   ^X01
    03A8  9198          VBUSDATA        <^X1E>,0                ;E6
01  03A9  9199          .BYTE   ^X01
    03AA  9200          VBUSDATA        <^X1E>,0                ;E7
01  03AB  9201          .BYTE   ^X01
    03AC  9202          VBUSDATA        <^X1E>,1                ;E8
01  03AD  9203          .BYTE   ^X01
    03AE  9204          VBUSDATA        <^X1E>,1                ;E9
01  03AF  9205          .BYTE   ^X01
    03B0  9206          VBUSDATA        <^X1E>,0                ;EA
01  03B1  9207          .BYTE   ^X01
    03B2  9208          VBUSDATA        <^X1E>,0                ;EB
01  03B3  9209          .BYTE   ^X01
    03B4  9210          VBUSDATA        <^X1E>,0                ;EC
01  03B5  9211          .BYTE   ^X01
    03B6  9212          VBUSDATA        <^X1E>,0                ;ED
01  03B7  9213          .BYTE   ^X01
    03B8  9214          VBUSDATA        <^X1E>,0                ;EE
01  03B9  9215          .BYTE   ^X01
    03BA  9216          VBUSDATA        <^X1E>,0                ;EF
01  03BB  9217          .BYTE   ^X01
    03BC  9218          VBUSDATA        <^X1E>,0                ;F0
01  03BD  9219          .BYTE   ^X01
    03BE  9220          VBUSDATA        <^X1E>,0                ;F1
01  03BF  9221          .BYTE   ^X01
    03C0  9222          VBUSDATA        <^X1E>,0                ;F2
01  03C1  9223          .BYTE   ^X01
    03C2  9224          VBUSDATA        <^X1E>,0                ;F3
01  03C3  9225          .BYTE   ^X01
    03C4  9226          VBUSDATA        <^X1E>,1                ;F4
01  03C5  9227          .BYTE   ^X01
```

```
                03C6  9228          VBUSDATA        <^X1E>,1                ;F5
01              03C7  9229          .BYTE   ^X01
                03C8  9230          VBUSDATA        <^X1E>,0                ;F6
01              03C9  9231          .BYTE   ^X01
                03CA  9232          VBUSDATA        <^X1E>,0                ;F7
01              03CB  9233          .BYTE   ^X01
                03CC  9234          VBUSDATA        <^X1E>,1                ;F8
01              03CD  9235          .BYTE   ^X01
                03CE  9236          VBUSDATA        <^X1E>,0                ;F9
01              03CF  9237          .BYTE   ^X01
                03D0  9238          VBUSDATA        <^X1E>,0                ;FA
01              03D1  9239          .BYTE   ^X01
                03D2  9240          VBUSDATA        <^X1E>,0                ;FB
01              03D3  9241          .BYTE   ^X01
                03D4  9242          VBUSDATA        <^X1E>,1                ;FC
01              03D5  9243          .BYTE   ^X01
                03D6  9244          VBUSDATA        <^X1E>,1                ;FD
01              03D7  9245          .BYTE   ^X01
                03D8  9246          VBUSDATA        <^X1E>,1                ;FE
01              03D9  9247          .BYTE   ^X01
                03DA  9248          VBUSDATA        <^X1E>,1                ;FF
                03DB  9249
                03DB  9250
                03DB  9251 ;
                03DB  9252 ; Expected VBUS values for Subtest 3
                03DB  9253 ;
                03DB  9254
01              03DB  9255 31$:     .BYTE   ^X01
                03DC  9256          VBUSDATA        <^X1E>,1                ;00
01              03DD  9257          .BYTE   ^X01
                03DE  9258          VBUSDATA        <^X1E>,1                ;01
01              03DF  9259          .BYTE   ^X01
                03E0  9260          VBUSDATA        <^X1E>,1                ;02
01              03E1  9261          .BYTE   ^X01
                03E2  9262          VBUSDATA        <^X1E>,1                ;03
01              03E3  9263          .BYTE   ^X01
                03E4  9264          VBUSDATA        <^X1E>,1                ;04
01              03E5  9265          .BYTE   ^X01
                03E6  9266          VBUSDATA        <^X1E>,1                ;05
01              03E7  9267          .BYTE   ^X01
                03E8  9268          VBUSDATA        <^X1E>,1                ;06
01              03E9  9269          .BYTE   ^X01
                03EA  9270          VBUSDATA        <^X1E>,1                ;07
01              03EB  9271          .BYTE   ^X01
                03EC  9272          VBUSDATA        <^X1E>,1                ;08
01              03ED  9273          .BYTE   ^X01
                03EE  9274          VBUSDATA        <^X1E>,1                ;09
01              03EF  9275          .BYTE   ^X01
                03F0  9276          VBUSDATA        <^X1E>,0                ;0A
01              03F1  9277          .BYTE   ^X01
                03F2  9278          VBUSDATA        <^X1E>,1                ;0B
01              03F3  9279          .BYTE   ^X01
                03F4  9280          VBUSDATA        <^X1E>,0                ;0C
01              03F5  9281          .BYTE   ^X01
                03F6  9282          VBUSDATA        <^X1E>,0                ;0D
```

```
                        01  03F7  9283          .BYTE   ^X01
                            03F8  9284          VBUSDATA        <^X1E>,0                ;0E
                        01  03F9  9285          .BYTE   ^X01
                            03FA  9286          VBUSDATA        <^X1E>,0                ;0F
                        01  03FB  9287          .BYTE   ^X01
                            03FC  9288          VBUSDATA        <^X1E>,1                ;10
                        01  03FD  9289          .BYTE   ^X01
                            03FE  9290          VBUSDATA        <^X1E>,1                ;11
                        01  03FF  9291          .BYTE   ^X01
                            0400  9292          VBUSDATA        <^X1E>,1                ;12
                        01  0401  9293          .BYTE   ^X01
                            0402  9294          VBUSDATA        <^X1E>,1                ;13
                        01  0403  9295          .BYTE   ^X01
                            0404  9296          VBUSDATA        <^X1E>,1                ;14
                        01  0405  9297          .BYTE   ^X01
                            0406  9298          VBUSDATA        <^X1E>,1                ;15
                        01  0407  9299          .BYTE   ^X01
                            0408  9300          VBUSDATA        <^X1E>,1                ;16
                        01  0409  9301          .BYTE   ^X01
                            040A  9302          VBUSDATA        <^X1E>,1                ;17
                        01  040B  9303          .BYTE   ^X01
                            040C  9304          VBUSDATA        <^X1E>,1                ;18
                        01  040D  9305          .BYTE   ^X01
                            040E  9306          VBUSDATA        <^X1E>,1                ;19
                        01  040F  9307          .BYTE   ^X01
                            0410  9308          VBUSDATA        <^X1E>,1                ;1A
                        01  0411  9309          .BYTE   ^X01
                            0412  9310          VBUSDATA        <^X1E>,1                ;1B
                        01  0413  9311          .BYTE   ^X01
                            0414  9312          VBUSDATA        <^X1E>,1                ;1C
                        01  0415  9313          .BYTE   ^X01
                            0416  9314          VBUSDATA        <^X1E>,1                ;1D
                        01  0417  9315          .BYTE   ^X01
                            0418  9316          VBUSDATA        <^X1E>,1                ;1E
                        01  0419  9317          .BYTE   ^X01
                            041A  9318          VBUSDATA        <^X1E>,1                ;1F
                        01  041B  9319          .BYTE   ^X01
                            041C  9320          VBUSDATA        <^X1E>,1                ;20
                        01  041D  9321          .BYTE   ^X01
                            041E  9322          VBUSDATA        <^X1E>,1                ;21
                        01  041F  9323          .BYTE   ^X01
                            0420  9324          VBUSDATA        <^X1E>,1                ;22
                        01  0421  9325          .BYTE   ^X01
                            0422  9326          VBUSDATA        <^X1E>,1                ;23
                        01  0423  9327          .BYTE   ^X01
                            0424  9328          VBUSDATA        <^X1E>,1                ;24
                        01  0425  9329          .BYTE   ^X01
                            0426  9330          VBUSDATA        <^X1E>,1                ;25
                        01  0427  9331          .BYTE   ^X01
                            0428  9332          VBUSDATA        <^X1E>,1                ;26
                        01  0429  9333          .BYTE   ^X01
                            042A  9334          VBUSDATA        <^X1E>,1                ;27
                        01  042B  9335          .BYTE   ^X01
                            042C  9336          VBUSDATA        <^X1E>,1                ;28
                        01  042D  9337          .BYTE   ^X01
```

```
                042E  9338          VBUSDATA        <^X1E>,1                ;29
            01  042F  9339          .BYTE   ^X01
                0430  9340          VBUSDATA        <^X1E>,1                ;2A
            01  0431  9341          .BYTE   ^X01
                0432  9342          VBUSDATA        <^X1E>,1                ;2B
            01  0433  9343          .BYTE   ^X01
                0434  9344          VBUSDATA        <^X1E>,1                ;2C
            01  0435  9345          .BYTE   ^X01
                0436  9346          VBUSDATA        <^X1E>,1                ;2D
            01  0437  9347          .BYTE   ^X01
                0438  9348          VBUSDATA        <^X1E>,1                ;2E
            01  0439  9349          .BYTE   ^X01
                043A  9350          VBUSDATA        <^X1E>,1                ;2F
            01  043B  9351          .BYTE   ^X01
                043C  9352          VBUSDATA        <^X1E>,1                ;30
            01  043D  9353          .BYTE   ^X01
                043E  9354          VBUSDATA        <^X1E>,1                ;31
            01  043F  9355          .BYTE   ^X01
                0440  9356          VBUSDATA        <^X1E>,1                ;32
            01  0441  9357          .BYTE   ^X01
                0442  9358          VBUSDATA        <^X1E>,0                ;33
            01  0443  9359          .BYTE   ^X01
                0444  9360          VBUSDATA        <^X1E>,1                ;34
            01  0445  9361          .BYTE   ^X01
                0446  9362          VBUSDATA        <^X1E>,1                ;35
            01  0447  9363          .BYTE   ^X01
                0448  9364          VBUSDATA        <^X1E>,1                ;36
            01  0449  9365          .BYTE   ^X01
                044A  9366          VBUSDATA        <^X1E>,1                ;37
            01  044B  9367          .BYTE   ^X01
                044C  9368          VBUSDATA        <^X1E>,1                ;38
            01  044D  9369          .BYTE   ^X01
                044E  9370          VBUSDATA        <^X1E>,1                ;39
            01  044F  9371          .BYTE   ^X01
                0450  9372          VBUSDATA        <^X1E>,1                ;3A
            01  0451  9373          .BYTE   ^X01
                0452  9374          VBUSDATA        <^X1E>,1                ;3B
            01  0453  9375          .BYTE   ^X01
                0454  9376          VBUSDATA        <^X1E>,1                ;3C
            01  0455  9377          .BYTE   ^X01
                0456  9378          VBUSDATA        <^X1E>,0                ;3D
            01  0457  9379          .BYTE   ^X01
                0458  9380          VBUSDATA        <^X1E>,0                ;3E
            01  0459  9381          .BYTE   ^X01
                045A  9382          VBUSDATA        <^X1E>,1                ;3F
            01  045B  9383          .BYTE   ^X01
                045C  9384          VBUSDATA        <^X1E>,0                ;40
            01  045D  9385          .BYTE   ^X01
                045E  9386          VBUSDATA        <^X1E>,0                ;41
            01  045F  9387          .BYTE   ^X01
                0460  9388          VBUSDATA        <^X1E>,0                ;42
            01  0461  9389          .BYTE   ^X01
                0462  9390          VBUSDATA        <^X1E>,0                ;43
            01  0463  9391          .BYTE   ^X01
                0464  9392          VBUSDATA        <^X1E>,0                ;44
```

```
01   0465  9393          .BYTE   ^X01
     0466  9394          VBUSDATA        <^X1E>,0                ;45
01   0467  9395          .BYTE   ^X01
     0468  9396          VBUSDATA        <^X1E>,0                ;46
01   0469  9397          .BYTE   ^X01
     046A  9398          VBUSDATA        <^X1E>,0                ;47
01   046B  9399          .BYTE   ^X01
     046C  9400          VBUSDATA        <^X1E>,1                ;48
01   046D  9401          .BYTE   ^X01
     046E  9402          VBUSDATA        <^X1E>,1                ;49
01   046F  9403          .BYTE   ^X01
     0470  9404          VBUSDATA        <^X1E>,1                ;4A
01   0471  9405          .BYTE   ^X01
     0472  9406          VBUSDATA        <^X1E>,1                ;4B
01   0473  9407          .BYTE   ^X01
     0474  9408          VBUSDATA        <^X1E>,1                ;4C
01   0475  9409          .BYTE   ^X01
     0476  9410          VBUSDATA        <^X1E>,1                ;4D
01   0477  9411          .BYTE   ^X01
     0478  9412          VBUSDATA        <^X1E>,1                ;4E
01   0479  9413          .BYTE   ^X01
     047A  9414          VBUSDATA        <^X1E>,0                ;4F
01   047B  9415          .BYTE   ^X01
     047C  9416          VBUSDATA        <^X1E>,0                ;50
01   047D  9417          .BYTE   ^X01
     047E  9418          VBUSDATA        <^X1E>,0                ;51
01   047F  9419          .BYTE   ^X01
     0480  9420          VBUSDATA        <^X1E>,0                ;52
01   0481  9421          .BYTE   ^X01
     0482  9422          VBUSDATA        <^X1E>,1                ;53
01   0483  9423          .BYTE   ^X01
     0484  9424          VBUSDATA        <^X1E>,0                ;54
01   0485  9425          .BYTE   ^X01
     0486  9426          VBUSDATA        <^X1E>,0                ;55
01   0487  9427          .BYTE   ^X01
     0488  9428          VBUSDATA        <^X1E>,1                ;56
01   0489  9429          .BYTE   ^X01
     048A  9430          VBUSDATA        <^X1E>,1                ;57
01   048B  9431          .BYTE   ^X01
     048C  9432          VBUSDATA        <^X1E>,0                ;58
01   048D  9433          .BYTE   ^X01
     048E  9434          VBUSDATA        <^X1E>,1                ;59
01   048F  9435          .BYTE   ^X01
     0490  9436          VBUSDATA        <^X1E>,1                ;5A
01   0491  9437          .BYTE   ^X01
     0492  9438          VBUSDATA        <^X1E>,1                ;5B
01   0493  9439          .BYTE   ^X01
     0494  9440          VBUSDATA        <^X1E>,0                ;5C
01   0495  9441          .BYTE   ^X01
     0496  9442          VBUSDATA        <^X1E>,0                ;5D
01   0497  9443          .BYTE   ^X01
     0498  9444          VBUSDATA        <^X1E>,0                ;5E
01   0499  9445          .BYTE   ^X01
     049A  9446          VBUSDATA        <^X1E>,0                ;5F
01   049B  9447          .BYTE   ^X01
```

```
                049C  9448          VBUSDATA        <^X1E>,0                ;60
     01         049D  9449          .BYTE   ^X01
                049E  9450          VBUSDATA        <^X1E>,0                ;61
     01         049F  9451          .BYTE   ^X01
                04A0  9452          VBUSDATA        <^X1E>,0                ;62
     01         04A1  9453          .BYTE   ^X01
                04A2  9454          VBUSDATA        <^X1E>,0                ;63
     01         04A3  9455          .BYTE   ^X01
                04A4  9456          VBUSDATA        <^X1E>,0                ;64
     01         04A5  9457          .BYTE   ^X01
                04A6  9458          VBUSDATA        <^X1E>,0                ;65
     01         04A7  9459          .BYTE   ^X01
                04A8  9460          VBUSDATA        <^X1E>,0                ;66
     01         04A9  9461          .BYTE   ^X01
                04AA  9462          VBUSDATA        <^X1E>,0                ;67
     01         04AB  9463          .BYTE   ^X01
                04AC  9464          VBUSDATA        <^X1E>,1                ;68
     01         04AD  9465          .BYTE   ^X01
                04AE  9466          VBUSDATA        <^X1E>,1                ;69
     01         04AF  9467          .BYTE   ^X01
                04B0  9468          VBUSDATA        <^X1E>,1                ;6A
     01         04B1  9469          .BYTE   ^X01
                04B2  9470          VBUSDATA        <^X1E>,1                ;6B
     01         04B3  9471          .BYTE   ^X01
                04B4  9472          VBUSDATA        <^X1E>,1                ;6C
     01         04B5  9473          .BYTE   ^X01
                04B6  9474          VBUSDATA        <^X1E>,1                ;6D
     01         04B7  9475          .BYTE   ^X01
                04B8  9476          VBUSDATA        <^X1E>,1                ;6E
     01         04B9  9477          .BYTE   ^X01
                04BA  9478          VBUSDATA        <^X1E>,1                ;6F
     01         04BB  9479          .BYTE   ^X01
                04BC  9480          VBUSDATA        <^X1E>,0                ;70
     01         04BD  9481          .BYTE   ^X01
                04BE  9482          VBUSDATA        <^X1E>,0                ;71
     01         04BF  9483          .BYTE   ^X01
                04C0  9484          VBUSDATA        <^X1E>,0                ;72
     01         04C1  9485          .BYTE   ^X01
                04C2  9486          VBUSDATA        <^X1E>,1                ;73
     01         04C3  9487          .BYTE   ^X01
                04C4  9488          VBUSDATA        <^X1E>,0                ;74
     01         04C5  9489          .BYTE   ^X01
                04C6  9490          VBUSDATA        <^X1E>,1                ;75
     01         04C7  9491          .BYTE   ^X01
                04C8  9492          VBUSDATA        <^X1E>,1                ;76
     01         04C9  9493          .BYTE   ^X01
                04CA  9494          VBUSDATA        <^X1E>,1                ;77
     01         04CB  9495          .BYTE   ^X01
                04CC  9496          VBUSDATA        <^X1E>,0                ;78
     01         04CD  9497          .BYTE   ^X01
                04CE  9498          VBUSDATA        <^X1E>,0                ;79
     01         04CF  9499          .BYTE   ^X01
                04D0  9500          VBUSDATA        <^X1E>,0                ;7A
     01         04D1  9501          .BYTE   ^X01
                04D2  9502          VBUSDATA        <^X1E>,0                ;7B
```

```
                    01  04D3  9503          .BYTE   ^X01
                        04D4  9504          VBUSDATA        <^X1E>,1                ;7C
                    01  04D5  9505          .BYTE   ^X01
                        04D6  9506          VBUSDATA        <^X1E>,0                ;7D
                    01  04D7  9507          .BYTE   ^X01
                        04D8  9508          VBUSDATA        <^X1E>,0                ;7E
                    01  04D9  9509          .BYTE   ^X01
                        04DA  9510          VBUSDATA        <^X1E>,1                ;7F
                    01  04DB  9511          .BYTE   ^X01
                        04DC  9512          VBUSDATA        <^X1E>,0                ;80
                    01  04DD  9513          .BYTE   ^X01
                        04DE  9514          VBUSDATA        <^X1E>,0                ;81
                    01  04DF  9515          .BYTE   ^X01
                        04E0  9516          VBUSDATA        <^X1E>,0                ;82
                    01  04E1  9517          .BYTE   ^X01
                        04E2  9518          VBUSDATA        <^X1E>,0                ;83
                    01  04E3  9519          .BYTE   ^X01
                        04E4  9520          VBUSDATA        <^X1E>,0                ;84
                    01  04E5  9521          .BYTE   ^X01
                        04E6  9522          VBUSDATA        <^X1E>,0                ;85
                    01  04E7  9523          .BYTE   ^X01
                        04E8  9524          VBUSDATA        <^X1E>,1                ;86
                    01  04E9  9525          .BYTE   ^X01
                        04EA  9526          VBUSDATA        <^X1E>,1                ;87
                    01  04EB  9527          .BYTE   ^X01
                        04EC  9528          VBUSDATA        <^X1E>,0                ;88
                    01  04ED  9529          .BYTE   ^X01
                        04EE  9530          VBUSDATA        <^X1E>,0                ;89
                    01  04EF  9531          .BYTE   ^X01
                        04F0  9532          VBUSDATA        <^X1E>,0                ;8A
                    01  04F1  9533          .BYTE   ^X01
                        04F2  9534          VBUSDATA        <^X1E>,0                ;8B
                    01  04F3  9535          .BYTE   ^X01
                        04F4  9536          VBUSDATA        <^X1E>,0                ;8C
                    01  04F5  9537          .BYTE   ^X01
                        04F6  9538          VBUSDATA        <^X1E>,0                ;8D
                    01  04F7  9539          .BYTE   ^X01
                        04F8  9540          VBUSDATA        <^X1E>,0                ;8E
                    01  04F9  9541          .BYTE   ^X01
                        04FA  9542          VBUSDATA        <^X1E>,0                ;8F
                    01  04FB  9543          .BYTE   ^X01
                        04FC  9544          VBUSDATA        <^X1E>,0                ;90
                    01  04FD  9545          .BYTE   ^X01
                        04FE  9546          VBUSDATA        <^X1E>,0                ;91
                    01  04FF  9547          .BYTE   ^X01
                        0500  9548          VBUSDATA        <^X1E>,0                ;92
                    01  0501  9549          .BYTE   ^X01
                        0502  9550          VBUSDATA        <^X1E>,0                ;93
                    01  0503  9551          .BYTE   ^X01
                        0504  9552          VBUSDATA        <^X1E>,1                ;94
                    01  0505  9553          .BYTE   ^X01
                        0506  9554          VBUSDATA        <^X1E>,1                ;95
                    01  0507  9555          .BYTE   ^X01
                        0508  9556          VBUSDATA        <^X1E>,1                ;96
                    01  0509  9557          .BYTE   ^X01
```

```
                050A  9558          VBUSDATA        <^X1E>,1                ;97
             01 050B  9559          .BYTE   ^X01
                050C  9560          VBUSDATA        <^X1E>,1                ;98
             01 050D  9561          .BYTE   ^X01
                050E  9562          VBUSDATA        <^X1E>,1                ;99
             01 050F  9563          .BYTE   ^X01
                0510  9564          VBUSDATA        <^X1E>,1                ;9A
             01 0511  9565          .BYTE   ^X01
                0512  9566          VBUSDATA        <^X1E>,1                ;9B
             01 0513  9567          .BYTE   ^X01
                0514  9568          VBUSDATA        <^X1E>,0                ;9C
             01 0515  9569          .BYTE   ^X01
                0516  9570          VBUSDATA        <^X1E>,0                ;9D
             01 0517  9571          .BYTE   ^X01
                0518  9572          VBUSDATA        <^X1E>,0                ;9E
             01 0519  9573          .BYTE   ^X01
                051A  9574          VBUSDATA        <^X1E>,1                ;9F
             01 051B  9575          .BYTE   ^X01
                051C  9576          VBUSDATA        <^X1E>,0                ;A0
             01 051D  9577          .BYTE   ^X01
                051E  9578          VBUSDATA        <^X1E>,0                ;A1
             01 051F  9579          .BYTE   ^X01
                0520  9580          VBUSDATA        <^X1E>,0                ;A2
             01 0521  9581          .BYTE   ^X01
                0522  9582          VBUSDATA        <^X1E>,0                ;A3
             01 0523  9583          .BYTE   ^X01
                0524  9584          VBUSDATA        <^X1E>,0                ;A4
             01 0525  9585          .BYTE   ^X01
                0526  9586          VBUSDATA        <^X1E>,0                ;A5
             01 0527  9587          .BYTE   ^X01
                0528  9588          VBUSDATA        <^X1E>,1                ;A6
             01 0529  9589          .BYTE   ^X01
                052A  9590          VBUSDATA        <^X1E>,1                ;A7
             01 052B  9591          .BYTE   ^X01
                052C  9592          VBUSDATA        <^X1E>,0                ;A8
             01 052D  9593          .BYTE   ^X01
                052E  9594          VBUSDATA        <^X1E>,0                ;A9
             01 052F  9595          .BYTE   ^X01
                0530  9596          VBUSDATA        <^X1E>,0                ;AA
             01 0531  9597          .BYTE   ^X01
                0532  9598          VBUSDATA        <^X1E>,0                ;AB
             01 0533  9599          .BYTE   ^X01
                0534  9600          VBUSDATA        <^X1E>,0                ;AC
             01 0535  9601          .BYTE   ^X01
                0536  9602          VBUSDATA        <^X1E>,0                ;AD
             01 0537  9603          .BYTE   ^X01
                0538  9604          VBUSDATA        <^X1E>,0                ;AE
             01 0539  9605          .BYTE   ^X01
                053A  9606          VBUSDATA        <^X1E>,0                ;AF
             01 053B  9607          .BYTE   ^X01
                053C  9608          VBUSDATA        <^X1E>,0                ;B0
             01 053D  9609          .BYTE   ^X01
                053E  9610          VBUSDATA        <^X1E>,0                ;B1
             01 053F  9611          .BYTE   ^X01
                0540  9612          VBUSDATA        <^X1E>,0                ;B2
```

```
01  0541  9613          .BYTE   ^X01
    0542  9614          VBUSDATA        <^X1E>,0                ;B3
01  0543  9615          .BYTE   ^X01
    0544  9616          VBUSDATA        <^X1E>,1                ;B4
01  0545  9617          .BYTE   ^X01
    0546  9618          VBUSDATA        <^X1E>,1                ;B5
01  0547  9619          .BYTE   ^X01
    0548  9620          VBUSDATA        <^X1E>,1                ;B6
01  0549  9621          .BYTE   ^X01
    054A  9622          VBUSDATA        <^X1E>,1                ;B7
01  054B  9623          .BYTE   ^X01
    054C  9624          VBUSDATA        <^X1E>,1                ;B8
01  054D  9625          .BYTE   ^X01
    054E  9626          VBUSDATA        <^X1E>,1                ;B9
01  054F  9627          .BYTE   ^X01
    0550  9628          VBUSDATA        <^X1E>,1                ;BA
01  0551  9629          .BYTE   ^X01
    0552  9630          VBUSDATA        <^X1E>,1                ;BB
01  0553  9631          .BYTE   ^X01
    0554  9632          VBUSDATA        <^X1E>,1                ;BC
01  0555  9633          .BYTE   ^X01
    0556  9634          VBUSDATA        <^X1E>,1                ;BD
01  0557  9635          .BYTE   ^X01
    0558  9636          VBUSDATA        <^X1E>,1                ;BE
01  0559  9637          .BYTE   ^X01
    055A  9638          VBUSDATA        <^X1E>,1                ;BF
01  055B  9639          .BYTE   ^X01
    055C  9640          VBUSDATA        <^X1E>,0                ;C0
01  055D  9641          .BYTE   ^X01
    055E  9642          VBUSDATA        <^X1E>,0                ;C1
01  055F  9643          .BYTE   ^X01
    0560  9644          VBUSDATA        <^X1E>,0                ;C2
01  0561  9645          .BYTE   ^X01
    0562  9646          VBUSDATA        <^X1E>,0                ;C3
01  0563  9647          .BYTE   ^X01
    0564  9648          VBUSDATA        <^X1E>,0                ;C4
01  0565  9649          .BYTE   ^X01
    0566  9650          VBUSDATA        <^X1E>,0                ;C5
01  0567  9651          .BYTE   ^X01
    0568  9652          VBUSDATA        <^X1E>,1                ;C6
01  0569  9653          .BYTE   ^X01
    056A  9654          VBUSDATA        <^X1E>,1                ;C7
01  056B  9655          .BYTE   ^X01
    056C  9656          VBUSDATA        <^X1E>,0                ;C8
01  056D  9657          .BYTE   ^X01
    056E  9658          VBUSDATA        <^X1E>,0                ;C9
01  056F  9659          .BYTE   ^X01
    0570  9660          VBUSDATA        <^X1E>,0                ;CA
01  0571  9661          .BYTE   ^X01
    0572  9662          VBUSDATA        <^X1E>,0                ;CB
01  0573  9663          .BYTE   ^X01
    0574  9664          VBUSDATA        <^X1E>,0                ;CC
01  0575  9665          .BYTE   ^X01
    0576  9666          VBUSDATA        <^X1E>,0                ;CD
01  0577  9667          .BYTE   ^X01
```

```
                0578  9668          VBUSDATA        <^X1E>,0                ;CE
01              0579  9669          .BYTE   ^X01
                057A  9670          VBUSDATA        <^X1E>,0                ;CF
01              057B  9671          .BYTE   ^X01
                057C  9672          VBUSDATA        <^X1E>,0                ;D0
01              057D  9673          .BYTE   ^X01
                057E  9674          VBUSDATA        <^X1E>,0                ;D1
01              057F  9675          .BYTE   ^X01
                0580  9676          VBUSDATA        <^X1E>,0                ;D2
01              0581  9677          .BYTE   ^X01
                0582  9678          VBUSDATA        <^X1E>,0                ;D3
01              0583  9679          .BYTE   ^X01
                0584  9680          VBUSDATA        <^X1E>,1                ;D4
01              0585  9681          .BYTE   ^X01
                0586  9682          VBUSDATA        <^X1E>,1                ;D5
01              0587  9683          .BYTE   ^X01
                0588  9684          VBUSDATA        <^X1E>,1                ;D6
01              0589  9685          .BYTE   ^X01
                058A  9686          VBUSDATA        <^X1E>,1                ;D7
01              058B  9687          .BYTE   ^X01
                058C  9688          VBUSDATA        <^X1E>,0                ;D8
01              058D  9689          .BYTE   ^X01
                058E  9690          VBUSDATA        <^X1E>,0                ;D9
01              058F  9691          .BYTE   ^X01
                0590  9692          VBUSDATA        <^X1E>,0                ;DA
01              0591  9693          .BYTE   ^X01
                0592  9694          VBUSDATA        <^X1E>,1                ;DB
01              0593  9695          .BYTE   ^X01
                0594  9696          VBUSDATA        <^X1E>,1                ;DC
01              0595  9697          .BYTE   ^X01
                0596  9698          VBUSDATA        <^X1E>,1                ;DD
01              0597  9699          .BYTE   ^X01
                0598  9700          VBUSDATA        <^X1E>,0                ;DE
01              0599  9701          .BYTE   ^X01
                059A  9702          VBUSDATA        <^X1E>,1                ;DF
01              059B  9703          .BYTE   ^X01
                059C  9704          VBUSDATA        <^X1E>,0                ;E0
01              059D  9705          .BYTE   ^X01
                059E  9706          VBUSDATA        <^X1E>,0                ;E1
01              059F  9707          .BYTE   ^X01
                05A0  9708          VBUSDATA        <^X1E>,0                ;E2
01              05A1  9709          .BYTE   ^X01
                05A2  9710          VBUSDATA        <^X1E>,0                ;E3
01              05A3  9711          .BYTE   ^X01
                05A4  9712          VBUSDATA        <^X1E>,0                ;E4
01              05A5  9713          .BYTE   ^X01
                05A6  9714          VBUSDATA        <^X1E>,0                ;E5
01              05A7  9715          .BYTE   ^X01
                05A8  9716          VBUSDATA        <^X1E>,0                ;E6
01              05A9  9717          .BYTE   ^X01
                05AA  9718          VBUSDATA        <^X1E>,0                ;E7
01              05AB  9719          .BYTE   ^X01
                05AC  9720          VBUSDATA        <^X1E>,1                ;E8
01              05AD  9721          .BYTE   ^X01
                05AE  9722          VBUSDATA        <^X1E>,1                ;E9
```

```
                        01  05AF  9723          .BYTE   ^X01
                            05B0  9724          VBUSDATA        <^X1E>,0                ;EA
                        01  05B1  9725          .BYTE   ^X01
                            05B2  9726          VBUSDATA        <^X1E>,0                ;EB
                        01  05B3  9727          .BYTE   ^X01
                            05B4  9728          VBUSDATA        <^X1E>,0                ;EC
                        01  05B5  9729          .BYTE   ^X01
                            05B6  9730          VBUSDATA        <^X1E>,0                ;ED
                        01  05B7  9731          .BYTE   ^X01
                            05B8  9732          VBUSDATA        <^X1E>,0                ;EE
                        01  05B9  9733          .BYTE   ^X01
                            05BA  9734          VBUSDATA        <^X1E>,0                ;EF
                        01  05BB  9735          .BYTE   ^X01
                            05BC  9736          VBUSDATA        <^X1E>,0                ;F0
                        01  05BD  9737          .BYTE   ^X01
                            05BE  9738          VBUSDATA        <^X1E>,0                ;F1
                        01  05BF  9739          .BYTE   ^X01
                            05C0  9740          VBUSDATA        <^X1E>,0                ;F2
                        01  05C1  9741          .BYTE   ^X01
                            05C2  9742          VBUSDATA        <^X1E>,0                ;F3
                        01  05C3  9743          .BYTE   ^X01
                            05C4  9744          VBUSDATA        <^X1E>,1                ;F4
                        01  05C5  9745          .BYTE   ^X01
                            05C6  9746          VBUSDATA        <^X1E>,1                ;F5
                        01  05C7  9747          .BYTE   ^X01
                            05C8  9748          VBUSDATA        <^X1E>,0                ;F6
                        01  05C9  9749          .BYTE   ^X01
                            05CA  9750          VBUSDATA        <^X1E>,0                ;F7
                        01  05CB  9751          .BYTE   ^X01
                            05CC  9752          VBUSDATA        <^X1E>,1                ;F8
                        01  05CD  9753          .BYTE   ^X01
                            05CE  9754          VBUSDATA        <^X1E>,0                ;F9
                        01  05CF  9755          .BYTE   ^X01
                            05D0  9756          VBUSDATA        <^X1E>,0                ;FA
                        01  05D1  9757          .BYTE   ^X01
                            05D2  9758          VBUSDATA        <^X1E>,0                ;FB
                        01  05D3  9759          .BYTE   ^X01
                            05D4  9760          VBUSDATA        <^X1E>,1                ;FC
                        01  05D5  9761          .BYTE   ^X01
                            05D6  9762          VBUSDATA        <^X1E>,1                ;FD
                        01  05D7  9763          .BYTE   ^X01
                            05D8  9764          VBUSDATA        <^X1E>,1                ;FE
                        01  05D9  9765          .BYTE   ^X01
                            05DA  9766          VBUSDATA        <^X1E>,1                ;FF
                            05DB  9767
                            05DB  9768 BEGIN_CPU_DATA
                            0002  9769
                            0002  9770 DCS_DATA
                            0001  9771
                            0001  9772 ; Micro-code to perform the instruction stream initialization and IRD1.
                            0001  9773
U 00, 7700,C800,0364,0300,0470,1801 0001  9774 ;% 00    NOP                             ; get the cpu up to speed
U 01, 5700,C840,0364,0308,0470,1802 000C  9778 ;% 01    R[TEMP2]_RDM                    ; load opcode in R[TEMP2]
U 02, 7700,F800,56D7,57FF,8470,1803 0017  9782 ;% 02    LONLIT_[52515000]               ; load opspecs in lonlit reg.
U 03, 7700,8800,5BE5,03D4,0470,1804 0022  9786 ;% 03    Q_R[LONLIT]                     ; move opspecs to Q register
```

```
U 04. 7700.0842.00A4.0308.0470.1805  002D  9790 ;% 04   R[TEMP2]_M[TEMP2].OR.Q                    ; OR opcode with opspecs
U 05. 7700.4868.5BE4.0308.0470.1806  0038  9794 ;% 05   M[TEMP8]_R[TEMP2]                        ; save in M[TEMP8] for error
U 06. 7700.C800.5BE4.03D8.4A70.1807  0043  9798 ;% 06   VA_R[ZERO]                              ; zero the VA register
U 07. 7701.C800.5BE4.0208.5590.1808  004E  9802 ;% 07   WRITE.LONG R[TEMP2]                     ; write data for XB0
U 08. 7700.0800.5BE4.0308.5470.1809  0059  9806 ;% 08   WDR_UNROT(R[TEMP2])                    ; validate cache location 0
U 09. 7700.C800.0364.0300.4470.180A  0064  9810 ;% 09   VA_VA+4                                ; update VA by 4
U 0A. 7700.3800.54D5.55D6.0470.180B  006F  9814 ;% 0A   LONLIT_[56555453]                      ; load opspecs in lonlit reg
U 0B. 7700.4869.5BE4.03D4.0470.180C  007A  9818 ;% 0B   M[TEMP9]_R[LONLIT]                     ; save in M[TEMP9] for error
U 0C. 7701.8800.5BE4.02D4.5590.180D  0085  9822 ;% 0C   WRITE.LONG R[LONLIT]                   ; write data for XB1
U 0D. 7700.C800.5BE4.03D4.5470.180E  0090  9826 ;% 0D   WDR_UNROT(R[LONLIT])                  ; validate cache location 4
U 0E. 7700.C800.5BE4.03D8.4870.180F  009B  9830 ;% 0E   PC_R[ZERO]                            ; zero PC to force prefetch
U 0F. 7700.C800.0364.1700.0470.1810  00A6  9834 ;% 0F   BUT/IRD1TST                           ; perform ird1
U 10. 7300.4800.0364.0300.0470.1811  00B1  9838 ;% 10   STOP.CLOCK                            ; halt
                                     00BC  9842
                                     00BC  9843 ; IRDX execution to trap generated address
                                     00BC  9844
U 11. 7700.4800.0364.0300.0470.1812  00BC  9845 ;% 11   NOP                                   ; get the cpu up to speed
U 12. 7600.8800.0364.0700.0470.1813  00C7  9849 ;% 12   STROBE.V.BUS,BUT/IRDX                 ; do irdx and strobe VBUS
U 13. 7300.4800.0364.0300.0470.1814  00D2  9853 ;% 13   STOP.CLOCK                            ; halt
                                     00DD  9857
                                     00DD  9858 ; FPA enable control and status check
                                     00DD  9859
U 14. 7700.C800.0364.0300.0470.1815  00DD  9860 ;% 14   NOP                                   ; get cpu up to speed
U 15. 5700.C800.0364.0300.2A70.1816  00E8  9864 ;% 15   WCTRL/FPA.ENABLE_WB5,WB_RDM           ; write the FPA enable bit
U 16. 7700.4800.0364.0300.0470.1817  00F3  9868 ;% 16   NOP                                   ; ...
U 17. 7700.4800.0364.C300.0470.1800  00FE  9872 ;% 17   BUT/NO.FPA,NEXT/1800                  ; check FPA status
U 18. 7300.C800.0364.0300.0470.1819  0109  9876 ;% 18   STOP.CLOCK                            ; halt
                                     0114  9880
                                     0114  9881 ; Control-P interrupt disable
                                     0114  9882
U 19. 7700.C800.0364.0300.0470.181A  0114  9883 ;% 19   NOP                                   ; get the cpu up to speed
U 1A. 7700.F800.7F9F.FFFF.8470.181B  011F  9887 ;% 1A   LONLIT_[00C00000]                     ; address of s&c bit in crar
U 1B. 7700.4800.5BE4.03D4.2470.181C  012A  9891 ;% 1B   WCTRL/LOADCRAR,WB_R[LONLIT]          ; address the s&c bit
U 1C. 7700.B800.7FDF.FFFF.8470.181D  0135  9895 ;% 1C   LONLIT_[00400000]                     ; data for ctrl-p int. disable
U 1D. 7700.4800.5BE4.03D4.2C70.181E  0140  9899 ;% 1D   WCTRL/CONWRITE,WB_R[LONLIT]          ; disable ctrl-p interrupts
U 1E. 7300.C800.0364.0300.0470.181F  014B  9903 ;% 1E   STOP.CLOCK                            ; halt
                                     0156  9907
                                     0156  9908 ; Control-P interrupt enable
                                     0156  9909
U 1F. 7700.C800.0364.0300.0470.1820  0156  9910 ;% 1F   NOP                                   ; get the cpu up to speed
U 20. 7700.F800.7F9F.FFFF.8470.1821  0161  9914 ;% 20   LONLIT_[00C00000]                     ; address of s&c bit in crar
U 21. 7700.C800.5BE4.03D4.2470.1822  016C  9918 ;% 21   WCTRL/LOADCRAR,WB_R[LONLIT]          ; address the s&c bit
U 22. 7700.7800.7FB7.FFFF.8470.1823  0177  9922 ;% 22   LONLIT_[00900000]                     ; data for ctrl-p int. enable
U 23. 7700.4800.5BE4.03D4.2C70.1824  0182  9926 ;% 23   WCTRL/CONWRITE,WB_R[LONLIT]          ; enable ctrl-p interrupts
U 24. 7300.C800.0364.0300.0470.1825  018D  9930 ;% 24   STOP.CLOCK                            ; halt
                                     0198  9934
                                     0198  9935 ; Set-up for RTEMP0 and RTEMP1 for error callout
                                     0198  9936
U 25. 7700.C800.0364.0300.0470.1826  0198  9937 ;% 25   NOP                                   ; get cpu up to speed
U 26. 7700.0848.5924.0300.0470.1827  01A3  9941 ;% 26   R[TEMP0]_M[TEMP8]                     ; move XB0 data to R[TEMP0]
U 27. 7700.0849.5924.0304.0470.1828  01AE  9945 ;% 27   R[TEMP1]_M[TEMP9]                     ; move XB1 data to R[TEMP1]
U 28. 7300.C800.0364.0300.0470.1829  01B9  9949 ;% 28   STOP.CLOCK                            ; halt
                                     01C4  9953
                                     01C4  9954 RTEMP_DATA
                                     0001  9955
```

```
                    00000000  0001  9956          .LONG   0                       ; rtemp0 = contents of xb0
                    00000000  0005  9957          .LONG   0                       ; rtemp1 = contents of xb1
                    00000000  0009  9958          .LONG   0                       ; rtemp2 = instruction
                              000D  9959
                              000D  9960 END_CPU_DATA
                              05DB  9961
                              05DB  9962 END_TEST_DATA
                              05DB
                              05DB        ;-----------------------------------------------------------------------------
                              05DB  9963
                              05DB  9964 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
0032        .SBTTL  "TEST   0CA     Second IRDX Instruction Address Decode Test 1
0032  9966 ;*****************************************************************************
0032  9967 ;++
0032  9968 ;
0032  9969 ; FUNCTIONAL DESCRIPTION
0032  9970 ;
0032  9971 ;       This test will examine the Native Mode IRDX roms when the IRD Counter
0032  9972 ;       is equal to one and the operand specifier is for a memory reference.
0032  9973 ;
0032  9974 ; TEST ALGORITHM
0032  9975 ;
0032  9976 ;       This test consists of three subtests.  The first subtest will examine
0032  9977 ;       the IRDX roms when the Floating Point Accelerator (FPA) is not present
0032  9978 ;       in the system.  The second subtest will examine the IRDX roms when the
0032  9979 ;       FPA is present but disabled.  The third subtest will examine the IRDX
0032  9980 ;       roms when the FPA is present and enabled.  The algorithm is as follows:
0032  9981 ;
0032  9982 ;        1.  Initialize the processor.
0032  9983 ;        2.  Disable CTRL/P interrupts.
0032  9984 ;        3.  Check to see if FPA present
0032  9985 ;               If FPA present
0032  9986 ;                   Then skip to Subtest 2
0032  9987 ;               Else fall through to subtest 1
0032  9988 ;
0032  9989 ;       Subtest 1
0032  9990 ;
0032  9991 ;        4.  Set-up flag to expect a micro trap.
0032  9992 ;        5.  Initialize loop I for 256 opcodes (00-FF).
0032  9993 ;        6.  Store the index I in a location that will be used to load into the
0032  9994 ;            WDREG.  The value is used as the opcode.
0032  9995 ;        7.  Loop on error from here.
0032  9996 ;        8.  Set-up DCS location 12 with a BUT/LOD.INC.BRA?,NEXT/1800.  This
0032  9997 ;            is used to get the IRD counter to 1 without sourcing an address
0032  9998 ;            from the IRDX roms (ie., by doing an IRDX).
0032  9999 ;        9.  Execute the following test microcode (performs the IRD1):
0032 10000 ;               a.  NOP to get the RDM up to speed.
0032 10001 ;               b.  Read the WDREG into RTEMP2 (opcode).
0032 10002 ;               c.  Load the operands into the LONLIT register.
0032 10003 ;               d.  Store the operands into the Q register to prepare for ORing
0032 10004 ;                   the opcode with the operands.
0032 10005 ;               e.  OR the opcode and operands together (XB0 data).
0032 10006 ;               f.  Store XB0 data in M[TEMP8] in case needed for error callout.
0032 10007 ;               g.  Zero the VA.
0032 10008 ;               h.  Validate cache location 0 by first writing the data in
0032 10009 ;                   RTEMP2 then reading the data back.
0032 10010 ;               i.  Update the VA by 4.
0032 10011 ;               j.  Load the second operand data into the LONLIT register
0032 10012 ;                   (XB1 data).
0032 10013 ;               k.  Store XB1 data in M[TEMP9] in case needed for error callout.
0032 10014 ;               l.  Validate cache location 4 by first writing the data in
0032 10015 ;                   the LONLIT register then reading the data back.
0032 10016 ;               m.  Load the PC with zero to cause a prefetch to take place.
0032 10017 ;               n.  Do an IRD1 to init the IRD counter and prepare for instruc-
0032 10018 ;                   tion/operand specifier evaluation.
0032 10019 ;               o.  Halt.
```

```
0032 10020 ;       10.  Perform the following test microcode (performs the LOD.INC.BRA?)
0032 10021 ;              a.  NOP to get the RDM up to speed.
0032 10022 ;              b.  Do an BUT/LOD.INC.BRA?,NEXT/1800
0032 10023 ;              c.  Halt.
0032 10024 ;       11.  Load DCS location 12 with a BUT/IRDX micro order.
0032 10025 ;       12.  Perform the following test microcode (performs the IRDX with the
0032 10026 ;            IRD counter = 1):
0032 10027 ;              a.  NOP to get the RDM up to speed.
0032 10028 ;              b.  Do an IRDX.
0032 10029 ;              c.  Halt.
0032 10030 ;       13.  Compare the contents of the CSTRP register (output from the IRDX
0032 10031 ;            Roms) against the Rom Table.
0032 10032 ;       14.  If the two do not equal, execute the following microcode and
0032 10033 ;            signal an error (this is done to set-up for printing RTEMP0 & 1
0032 10034 ;            on the error callout):
0032 10035 ;              a.  NOP to get the RDM up to speed.
0032 10036 ;              b.  Load RTEMP0 with the contents of M[TEMP8]
0032 10037 ;              c.  Load RTEMP1 with the contents of M[TEMP9]
0032 10038 ;              d.  Halt
0032 10039 ;       15.  Continue with loop I for the 256 VAX opcodes.
0032 10040 ;       16.  Re-enable CTRL/P interrupts then skip to next tests.
0032 10041 ;
0032 10042 ;       Subtest 2
0032 10043 ;
0032 10044 ;       17.  Load the FPA disable bit (bit <5> equal to 0).
0032 10045 ;       18.  Loop on error from here.
0032 10046 ;       19.  Set-up to expect micro trap.
0032 10047 ;       20.  Execute the test microcode.  This is done only to get the FPA to
0032 10048 ;            a state where it can be disabled or enabled.  If this had not been
0032 10049 ;            done before then the FPA would be setting in the initialize state
0032 10050 ;            which forces FPA enable.
0032 10051 ;       21.  Execute the following microcode to disable the FPA:
0032 10052 ;              a.  NOP to get the RDM up to speed.
0032 10053 ;              b.  Read the RDM WDREG onto the W Bus and then write the W Bus
0032 10054 ;                  to the FPA.
0032 10055 ;              c.  Perform a NO.FPA test to insure that the FPA got disabled.
0032 10056 ;              d.  Halt.
0032 10057 ;       22.  Check the Trap register to insure that the FPA got disabled.
0032 10058 ;       23.  If the FPA is not disabled, then signal an error.
0032 10059 ;       24.  Set-up flag to expect a micro trap.
0032 10060 ;       25.  Initialize loop I for 256 opcodes (00-FF).
0032 10061 ;       26.  Store the index I in a location that will be used to load into the
0032 10062 ;            WDREG.  The value is used as the opcode.
0032 10063 ;       27.  Loop on error from here.
0032 10064 ;       28.  Set-up DCS location 12 with a BUT/LOD.INC.BRA?,NEXT/1800.  This
0032 10065 ;            is used to get the IRD counter to 1 without sourcing an address
0032 10066 ;            from the IRDX roms (ie., by doing an IRDX).
0032 10067 ;       29.  Execute the following test microcode:
0032 10068 ;              a.  NOP to get the RDM up to speed.
0032 10069 ;              b.  Read the WDREG into RTEMP2 (opcode).
0032 10070 ;              c.  Load the operands into the LONLIT register.
0032 10071 ;              d.  Store the operands into the Q register to prepare for ORing
0032 10072 ;                  the opcode with the operands.
0032 10073 ;              e.  OR the opcode and operands together (XB0 data).
0032 10074 ;              f.  Store XB0 data in M[TEMP8] in case needed for error callout.
```

```
0032 10075 ;                    g.  Zero the VA.
0032 10076 ;                    h.  Validate cache location 0 by first writing the data in
0032 10077 ;                        RTEMP2 then reading the data back.
0032 10078 ;                    i.  Update the VA by 4.
0032 10079 ;                    j.  Load the second operand data into the LONLIT register
0032 10080 ;                        (XB1 data).
0032 10081 ;                    k.  Store XB1 data in M[TEMP9] in case needed for error callout.
0032 10082 ;                    l.  Validate cache location 4 by first writing the data in
0032 10083 ;                        the LONLIT register then reading the data back.
0032 10084 ;                    m.  Load the PC with zero to cause a prefetch to take place.
0032 10085 ;                    n.  Do an IRD1 to init the IRD counter and prepare for instruc-
0032 10086 ;                        tion/operand specifier evaluation.
0032 10087 ;                    o.  Halt.
0032 10088 ;            30.  Perform the following test microcode.
0032 10089 ;                    a.  NOP to get the RDM up to speed.
0032 10090 ;                    b.  Do an BUT/LOD.INC.BRA?,NEXT/1800
0032 10091 ;                    c.  Halt.
0032 10092 ;            31.  Load DCS location 12 with a BUT/IRDX micro order.
0032 10093 ;            32.  Perform the following test microcode (performs the IRDX with the
0032 10094 ;                 IRD counter = 1):
0032 10095 ;                    a.  NOP to get the RDM up to speed.
0032 10096 ;                    b.  Do an IRDX.
0032 10097 ;                    c.  Halt.
0032 10098 ;            33.  Compare the contents of the CSTRP register (output from the IRDX
0032 10099 ;                 Roms) against the Rom Table.
0032 10100 ;            34.  If the two do not equal, execute the following microcode and
0032 10101 ;                 signal an error (this is done to set-up for printing RTEMP0 & 1
0032 10102 ;                 on the error callout):
0032 10103 ;                    a.  NOP to get the RDM up to speed.
0032 10104 ;                    b.  Load RTEMP0 with the contents of M[TEMP8]
0032 10105 ;                    c.  Load RTEMP1 with the contents of M[TEMP9]
0032 10106 ;                    d.  Halt
0032 10107 ;            35.  Continue with loop I for the 256 VAX opcodes.
0032 10108 ;            36.  Go to Subtest 3.
0032 10109 ;
0032 10110 ;            Subtest 3
0032 10111 ;
0032 10112 ;            37.  Load the FPA enable bit (bit <5> equal to 1).
0032 10113 ;            38.  Loop on error from here.
0032 10114 ;            39.  Set-up to expect micro trap.
0032 10115 ;            40.  Execute the following microcode to enable the FPA:
0032 10116 ;                    a.  NOP to get the RDM up to speed.
0032 10117 ;                    b.  Read the RDM WDREG onto the W Bus and then write the W Bus
0032 10118 ;                        to the FPA.
0032 10119 ;                    c.  Perform a NO.FPA test to insure that the FPA got enabled.
0032 10120 ;                    d.  Halt.
0032 10121 ;            41.  Check the Trap register to insure that the FPA got enabled.
0032 10122 ;            42.  If the FPA is not enabled, then signal an error.
0032 10123 ;            43.  Set-up flag to expect a micro trap.
0032 10124 ;            44.  Initialize loop I for 256 opcodes (00-FF).
0032 10125 ;            45.  Store the index I in a location that will be used to load into the
0032 10126 ;                 WDREG.  The value is used as the opcode.
0032 10127 ;            46.  Loop on error from here.
0032 10128 ;            47.  Set then Clear the DCLO bit in the RDCTRL register to perform an
0032 10129 ;                 initialize.  This is done because when the test microcode
```

```
0032 10130 ;            executes, if the opcode was an FP opcode then there is a
0032 10131 ;            possibility that the processor could be setting there in an
0032 10132 ;            FPAWAIT state (M clock and D clock stalled).  If this were to
0032 10133 ;            happen then the cpu clocks would no longer run and the remaining
0032 10134 ;            test opcodes would fail.
0032 10135 ;       48.  Set-up DCS location 12 with a BUT/LOD.INC.BRA?,NEXT/1800.  This
0032 10136 ;            is used to get the IRD counter to 1 without sourcing an address
0032 10137 ;            from the IRDX roms (ie., by doing an IRDX).
0032 10138 ;       49.  Execute the following test microcode:
0032 10139 ;               a.  NOP to get the RDM up to speed.
0032 10140 ;               b.  Read the WDREG into RTEMP2 (opcode).
0032 10141 ;               c.  Load the operands into the LONLIT register.
0032 10142 ;               d.  Store the operands into the Q register to prepare for ORing
0032 10143 ;                   the opcode with the operands.
0032 10144 ;               e.  OR the opcode and operands together (XB0 data).
0032 10145 ;               f.  Store XB0 data in M[TEMP8] in case needed for error callout.
0032 10146 ;               g.  Zero the VA.
0032 10147 ;               h.  Validate cache location 0 by first writing the data in
0032 10148 ;                   RTEMP2 then reading the data back.
0032 10149 ;               i.  Update the VA by 4.
0032 10150 ;               j.  Load the second operand data into the LONLIT register
0032 10151 ;                   (XB1 data).
0032 10152 ;               k.  Store XB1 data in M[TEMP9] in case needed for error callout.
0032 10153 ;               l.  Validate cache location 4 by first writing the data in
0032 10154 ;                   the LONLIT register then reading the data back.
0032 10155 ;               m.  Load the PC with zero to cause a prefetch to take place.
0032 10156 ;               n.  Do an IRD1 to init the IRD counter and prepare for instruc-
0032 10157 ;                   tion/operand specifier evaluation.
0032 10158 ;               o.  Halt.
0032 10159 ;       50.  Perform the following test microcode.
0032 10160 ;               a.  NOP to get the RDM up to speed.
0032 10161 ;               b.  Do an BUT/LOD.INC.BRA?,NEXT/1800
0032 10162 ;               c.  Halt.
0032 10163 ;       51.  Load DCS location 12 with a BUT/IRDX micro order.
0032 10164 ;       52.  Perform the following test microcode (performs the IRDX with the
0032 10165 ;            IRD counter = 1):
0032 10166 ;               a.  NOP to get the RDM up to speed.
0032 10167 ;               b.  Do an IRDX.
0032 10168 ;               c.  Halt.
0032 10169 ;       53.  Compare the contents of the CSTRP register (output from the IRDX
0032 10170 ;            Roms) against the Rom Table.
0032 10171 ;       54.  If the two do not equal, execute the following microcode and
0032 10172 ;            signal an error (this is done to set-up for printing RTEMP0 & 1
0032 10173 ;            on the error callout):
0032 10174 ;               a.  NOP to get the RDM up to speed.
0032 10175 ;               b.  Load RTEMP0 with the contents of M[TEMP8]
0032 10176 ;               c.  Load RTEMP1 with the contents of M[TEMP9]
0032 10177 ;               d.  Halt
0032 10178 ;       55.  Continue with loop I for the 256 VAX opcodes.
0032 10179 ;       56.  Re-enable CTRL/P interrupts.
0032 10180 ;       57.  Skip to next tests.
0032 10181 ;
0032 10182 ; TEST PATTERNS
0032 10183 ;
0032 10184 ;            For this test the opcodes are generated by a simple count pattern.  The
```

```
0032 10185 ;       opcodes are based on the 256 base machine instructions as specified in
0032 10186 ;       the VAX architecture handbook.  The generated opcodes are written to
0032 10187 ;       the CPU and combined (ORed) with three bytes of operand data to form
0032 10188 ;       a machine instruction.  The operands all specify Register Deferred mode
0032 10189 ;       using registers 0-6 and are loaded from the LONLIT register.
0032 10190 ;
0032 10191 ;       Also required are two tables of expected control store addresses for
0032 10192 ;       the second IRDX cycle.  One table contains the expected addresses for
0032 10193 ;       the second IRDX when the FPA is disabled, the other for when it is
0032 10194 ;       enabled.
0032 10195 ;
0032 10196 ; LOGIC DESCRIPTION
0032 10197 ;
0032 10198 ;       This test will check the validity of the IRDX roms on the DPM module.
0032 10199 ;       The IRDX roms are addressed by bits <7:0> of the Instruction Register.
0032 10200 ;       Along with being addressed by the IR, there are three other signals
0032 10201 ;       that affect the address.  They are IRD CTR 0 H, FPA ENABLED L, and
0032 10202 ;       REG MODE H.  This test will examine every 8th(hex) address starting at
0032 10203 ;       address 6 with FPA disabled and starting at address 4 with FPA enabled.
0032 10204 ;
0032 10205 ; ASSUMPTIONS
0032 10206 ;
0032 10207 ;       This test assumes the following logic has already been tested; the
0032 10208 ;       CPU's data path, the cache, the XB, and the IR and OSR.
0032 10209 ;
0032 10210 ; ERROR DESCRIPTION
0032 10211 ;
0032 10212 ;       Subtest 1
0032 10213 ;
0032 10214 ;               Expected CS Address
0032 10215 ;               Received CS Address
0032 10216 ;
0032 10217 ;       Subtest 2
0032 10218 ;
0032 10219 ;               First IFERROR
0032 10220 ;
0032 10221 ;                       Expected CS Address
0032 10222 ;                       Received CS Address
0032 10223 ;
0032 10224 ;               Second IFERROR
0032 10225 ;
0032 10226 ;                       Expected IRDX Rom Contents
0032 10227 ;                       Received IRDX Rom Contents
0032 10228 ;                       Loop I count
0032 10229 ;                       Contents of XB0
0032 10230 ;                       Contents of XB1
0032 10231 ;
0032 10232 ;       Subtest 3
0032 10233 ;
0032 10234 ;               First IFERROR
0032 10235 ;
0032 10236 ;                       Expected CS Address
0032 10237 ;                       Received CS Address
0032 10238 ;
0032 10239 ;               Second IFERROR
```

```
                        0032 10240 ;
                        0032 10241 ;                    Expected IRDX Rom Contents
                        0032 10242 ;                    Received IRDX Rom Contents
                        0032 10243 ;                    Loop I count
                        0032 10244 ;                    Contents of XB0
                        0032 10245 ;                    Contents of XB1
                        0032 10246 ;--
                        0032 10247 ;*****************************************************************************
                        0032 10248
                        0032 10249          INITIALIZE                          ; initialize the comet cpu
                        0034 10250          FETCH           <^X19>              ; disable ctrl-p interrupts
                        0039 10251          BRSTCLK         <^X1E>              ; ...
                        003D 10252          FETCH           <^X16>              ; execute a but/no.fpa to check
                        0042 10253          BRSTCLK         <^X18>              ; for fpa present
                        0046 10254          CMPREGMSK       CSTRP,1$,,2$,,W     ; is the fpa present?
                        0052 10255          SKIP            100$,ONERROR        ; if so then skip to subtest 2
                        0059 10256
                        0059 10257          SUBTEST         1
                        005C
                        005C          ;////////////////////////////////////////////////////////////////////////////
                        005C 10258 ;+
                        005C 10259 ;
                        005C 10260 ; FPA is not present.
                        005C 10261 ;
                        005C 10262 ;-
                        005C 10263          EXPUTRAP                            ; else expect non-dcs addresses
                        005E 10264          LOOP            I,1,256             ; loop for 256 opcodes
                        0065 10265          SAVEINDEX       I,9$                ; save the loop count
                        006A 10266          LOADREG         WDREG,9$,,L         ; load wdreg with loop count
                        0072 10267          ERRLOOP                             ; expect micro trap
                        0074 10268          LOADDCS         20$,,<^X12>,1       ; load BUT/LOD.INC.BRA?
                        007B 10269          FETCH           0                   ; execute ird1tst
                        0080 10270          BRSTCLK         <^X10>,INH          ; ...
                        0084 10271          FETCH           <^X11>              ; execute BUT/LOD.INC.BRA?
                        0089 10272          BRSTCLK         <^X13>,INH          ; ...
                        008D 10273          LOADDCS         21$,,<^X12>,1       ; load BUT/IRDX
                        0094 10274          FETCH           <^X11>              ; execute the second irdx
                        0099 10275          BRSTCLK         <^X13>,INH          ; ...
                        009D 10276          CMPREGMSK       CSTRP,30$,I,6$,,W   ; is the cs address correct?
                        00A9 10277          SKIP            50$,NOERROR         ; continue testing if no error
                        00B0 10278          FETCH           <^X25>              ; start dcs program
                        00B5 10279          BRSTCLK         <^X28>              ; end dcs program
                        00B9 10280          IFERROR         2,<<IRD>>,<<MIC>>   ; signal an error
                        00C0 10281 50$:     ENDLOOP         I                   ; continue loop
                        00C3 10282          FETCH           <^X1F>              ; re-enable ctrl-p interrupts
                        00C8 10283          BRSTCLK         <^X24>              ; ...
                        00CC 10284          SKIP                                ; skip on to the next test
                        00D3 10285
                        00D3 10286 100$:    SUBTEST         2
                        00D6
                        00D6          ;////////////////////////////////////////////////////////////////////////////
                        00D6 10287 ;+
                        00D6 10288 ;
                        00D6 10289 ; FPA present but not enabled
                        00D6 10290 ;
```

```
                     00D6 10291 ;-
                     00D6 10292          LOADREG         WDREG,7$,,L             ; load fpa disable bit
                     00DE 10293          ERRLOOP                                 ; loop on error here
                     00E0 10294          EXPUTRAP                                ; expect micro trap
                     00E2 10295          FETCH           0                       ; start dcs program
                     00E7 10296          BRSTCLK         <^X10>,INH              ; end dcs program
                     00EB 10297          FETCH           <^X14>                  ; execute the disable
                     00F0 10298          BRSTCLK         <^X18>,INH              ; and read back fpa status
                     00F4 10299          CMPREGMSK       CSTRP,3$,,5$,,W         ; is the fpa disabled?
                     0100 10300          IFERROR         ,,<FPA>                 ; if not, signal an error
                     0106 10301          EXPUTRAP                                ; else expect non-dcs addresses
                     0108 10302          LOOP            I,1,256                 ; loop for 256 opcodes
                     010F 10303          SAVEINDEX       I,9$                    ; save the loop count
                     0114 10304          LOADREG         WDREG,9$,,L             ; load wdreg with loop count
                     011C 10305          ERRLOOP                                 ; loop on error from here
                     011E 10306          LOADDCS         20$,,<^X12>,1           ; load BUT/LOD.INC.BRA?
                     0125 10307          FETCH           0                       ; execute ird1tst
                     012A 10308          BRSTCLK         <^X10>,INH              ; ...
                     012E 10309          FETCH           <^X11>                  ; execute BUT/LOD.INC.BRA?
                     0133 10310          BRSTCLK         <^X13>,INH              ; ...
                     0137 10311          LOADDCS         21$,,<^X12>,1           ; load BUT/IRDX
                     013E 10312          FETCH           <^X11>                  ; execute the second irdx
                     0143 10313          BRSTCLK         <^X13>,INH              ; ...
                     0147 10314          CMPREGMSK       CSTRP,30$,I,6$,,W       ; is the cs address correct?
                     0153 10315          SKIP            51$,NOERROR             ; continue testing if no error
                     015A 10316          FETCH           <^X25>                  ; start dcs program
                     015F 10317          BRSTCLK         <^X28>                  ; end dcs program
                     0163 10318          IFERROR         2,<<IRD>>,<<MIC>>       ; signal an error
                     016A 10319 51$:     ENDLOOP         I                       ; continue loop
                     016D 10320
                     016D 10321          SUBTEST         3
                     0170
                     0170       ;////////////////////////////////////////////////////////////////////////////////
                     0170 10322 ;+
                     0170 10323 ;
                     0170 10324 ; FPA present and enabled
                     0170 10325 ;
                     0170 10326 ;-
                     0170 10327          LOADREG         WDREG,8$,,L             ; load fpa enable bit
                     0178 10328          ERRLOOP                                 ; loop on error from here
                     017A 10329          FETCH           <^X14>                  ; enable the fpa
                     017F 10330          BRSTCLK         <^X18>,INH              ; and read back fpa enable
                     0183 10331          CMPREGMSK       CSTRP,4$,,5$,,W         ; is the fpa enabled?
                     018F 10332          IFERROR         ,,<FPA>                 ; if not, signal an error
                     0195 10333          EXPUTRAP                                ; else expect non-dcs addresses
                     0197 10334          LOOP            I,1,256                 ; loop for 256 opcodes
                     019E 10335          SAVEINDEX       I,9$                    ; save the loop count
                     01A3 10336          LOADREG         WDREG,9$,,L             ; load wdreg with loop count
                     01AB 10337          ERRLOOP                                 ; loop on error from here
                     01AD 10338          LOADDCS         20$,,<^X12>,1           ; load BUT/LOD.INC.BRA?
                     01B4 10339          LOADREG         RDCTRL,10$,,B           ; assert dclo
                     01BC 10340          LOADREG         RDCTRL,11$,,B           ; de-assert dclo
                     01C4 10341          FETCH           0                       ; execute ird1tst
                     01C9 10342          BRSTCLK         <^X10>,INH              ; ...
                     01CD 10343          FETCH           <^X11>                  ; execute BUT/LOD.INC.BRA?
```

```
                     01D2 10344          BRSTCLK         <^X13>,INH                      ; ...
                     01D6 10345          LOADDCS         21$,,<^X12>,1                   ; load BUT/IRDX
                     01DD 10346          FETCH           <^X11>                          ; execute the second irdx
                     01E2 10347          BRSTCLK         <^X13>,INH                      ; ...
                     01E6 10348          CMPREGMSK       CSTRP,31$,I,6$,,W               ; is the cs address correct?
                     01F2 10349          SKIP            52$,NOERROR                     ; continue testing if no error
                     01F9 10350          FETCH           <^X25>                          ; start dcs program
                     01FE 10351          BRSTCLK         <^X28>                          ; end dcs program
                     0202 10352          IFERROR         2,<<IRD>>,<<MIC>>               ; signal an error
                     0209 10353 52$:     ENDLOOP         I                               ; continue loop
                     020C 10354          FETCH           <^X1F>                          ; re-enable ctrl-p interrupts
                     0211 10355          BRSTCLK         <^X24>                          ; ...
                     0215 10356          SKIP                                            ; end of test
                     021C 10357
                     021C 10358 BEGIN_TEST_DATA
                     021C
                     021C        ;-----------------------------------------------------------------------------
                     021C 10359
                1802 021C 10360 1$:      .WORD   ^X1802
                3FFE 021E 10361 2$:      .WORD   ^X3FFE
                     0220 10362
                1801 0220 10363 3$:      .WORD   ^X1801
                1800 0222 10364 4$:      .WORD   ^X1800
                3FFD 0224 10365 5$:      .WORD   ^X3FFD
                     0226 10366
                3FFF 0226 10367 6$:      .WORD   ^X3FFF
                     0228 10368
            00000000 0228 10369 7$:      .LONG   0
            00000020 022C 10370 8$:      .LONG   ^X00000020
                     0230 10371
            00000000 0230 10372 9$:      .LONG   0
                     0234 10373
                  40 0234 10374 10$:     .BYTE   DC_LOW
                  00 0235 10375 11$:     .BYTE   ^X00
                     0236 10376
                     0236 10377 ; Loadable microcode
                     0236 10378
                     0236 10379 20$:
U 12, 7700,4800,0364,1B00,0470,1800  0236 10380 ;% 12   BUT/LOD.INC.BRA,NEXT/1800          ; load OSR & increment IRD cntr
                     0241 10384 21$:
U 12, 7700,8800,0364,0700,0470,1813  0241 10385 ;% 12   BUT/IRDX                         ; do irdx
                     024C 10389
                     024C 10390 ;
                     024C 10391 ; Expected CS Addresses for Subtests 1 and 2
                     024C 10392 ;
                     024C 10393 ; The following is a table of all the expected control store addresses for the
                     024C 10394 ; second IRDX.
                     024C 10395 ;
 03F8 03F8 03F8 03F8 024C 10396 30$:     .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 00 - 06
 0332 032A 0224 0318 025A 10397          .WORD   ^X03F8,^X02A4,^X02A8,^X0318,^X0224,^X032A,^X0332 ; 07 - 0D
 03F8 03F8 03F8 03F8 0268 10398          .WORD   ^X03AA,^X02A2,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 0E - 14
 03F8 03F8 03F8 03F8 0276 10399          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 15 - 1B
 027C 0294 027C 03F8 0284 10400          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X027C,^X0294,^X027C ; 1C - 22
 0302 0302 0294 0368 0292 10401          .WORD   ^X0294,^X0360,^X0294,^X0368,^X0294,^X0302,^X0302 ; 23 - 29
 03F8 03B3 03B3 03B1 02A0 10402          .WORD   ^X0274,^X0274,^X03B1,^X03B1,^X03B3,^X03B3,^X03F8 ; 2A - 30
```

```
027C 03BB 0312 0312 00AE 0051 03F8  02AE 10403          .WORD   ^X03F8,^X0051,^X00AE,^X0312,^X0312,^X03BB,^X027C ; 31 - 37
00EE 028F 00EE 030A 030A 0320 02BC  02BC 10404          .WORD   ^X02BC,^X0320,^X030A,^X030A,^X00EE,^X028F,^X00EE ; 38 - 3E
0081 0081 0351 0351 0088 0088 03F8  02CA 10405          .WORD   ^X03F8,^X0088,^X0088,^X0351,^X0351,^X0081,^X0081 ; 3F - 45
0339 00CC 00C6 00C6 00C6 0095 0095  02D8 10406          .WORD   ^X0095,^X0095,^X00C6,^X00C6,^X00C6,^X00CC,^X0339 ; 46 - 4C
03F8 031D 00A6 030D 00E0 00BC 0339  02E6 10407          .WORD   ^X0339,^X00BC,^X00E0,^X030D,^X00A6,^X031D,^X03F8 ; 4D - 53
03F8 03F8 024E 03F8 00AD 01E1 01D2  02F4 10408          .WORD   ^X01D2,^X01E1,^X00AD,^X03F8,^X024E,^X03F8,^X03F8 ; 54 - 5A
00A2 00A2 02CA 02BA 02B2 02AC 03F8  0302 10409          .WORD   ^X03F8,^X02AC,^X02B2,^X02BA,^X02CA,^X00A2,^X00A2 ; 5B - 61
033D 00E8 00E8 00C0 00C0 00A2 00A2  0310 10410          .WORD   ^X00A2,^X00A2,^X00C0,^X00C0,^X00E8,^X00E8,^X033D ; 62 - 68
00A0 0339 0339 0339 033D 033D 033D  031E 10411          .WORD   ^X033D,^X033D,^X033D,^X0339,^X0339,^X0339,^X00A0 ; 69 - 6F
0341 035B 02C0 03F8 0315 0335 0315  032C 10412          .WORD   ^X0315,^X0335,^X0315,^X03F8,^X02C0,^X035B,^X0341 ; 70 - 76
01E5 03F8 0064 0004 0092 0092 03F8  033A 10413          .WORD   ^X03F8,^X0092,^X0092,^X0004,^X0064,^X03F8,^X01E5 ; 77 - 7D
0297 006F 025B 004F 023D 03F8 01EE  0348 10414          .WORD   ^X01EE,^X03F8,^X023D,^X004F,^X025B,^X006F,^X0297 ; 7E - 84
00CF 0291 00AF 028B 008C 0309 0040  0356 10415          .WORD   ^X0040,^X0309,^X008C,^X028B,^X00AF,^X0291,^X00CF ; 85 - 8B
01AF 0052 00EE 02AF 01CF 008F 0279  0364 10416          .WORD   ^X0279,^X008F,^X01CF,^X02AF,^X00EE,^X0052,^X01AF ; 8C - 92
0051 0051 03F8 03F8 03F8 03F8 000A  0372 10417          .WORD   ^X000A,^X03F8,^X03F8,^X03F8,^X03F8,^X0051,^X0051 ; 93 - 99
023D 03F8 00EE 028F 000F 00EE 00EE  0380 10418          .WORD   ^X00EE,^X00EE,^X000F,^X028F,^X00EE,^X03F8,^X023D ; 9A - A0
008C 0309 0040 0297 006F 025B 004F  038E 10419          .WORD   ^X004F,^X025B,^X006F,^X0297,^X0040,^X0309,^X008C ; A1 - A7
01CF 008F 0279 00CF 0291 00AF 028B  039C 10420          .WORD   ^X028B,^X00AF,^X0291,^X00CF,^X0279,^X008F,^X01CF ; A8 - AE
03F8 03F8 000A 01AF 0052 00EE 02AF  03AA 10421          .WORD   ^X02AF,^X00EE,^X0052,^X01AF,^X000A,^X03F8,^X03F8 ; AF - B5
03F8 03F8 03F8 03F8 03F8 03F8 03F8  03B8 10422          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; B6 - BC
006F 025B 004F 023D 03F8 03F8 03F8  03C6 10423          .WORD   ^X03F8,^X03F8,^X03F8,^X023D,^X004F,^X025B,^X006F ; BD - C3
0291 00AF 028B 008C 0309 0040 0297  03D4 10424          .WORD   ^X0297,^X0040,^X0309,^X008C,^X028B,^X00AF,^X0291 ; C4 - CA
0052 00EE 02AF 01CF 008F 0279 00CF  03E2 10425          .WORD   ^X00CF,^X0279,^X008F,^X01CF,^X02AF,^X00EE,^X0052 ; CB - D1
0231 03F8 03F8 03F8 03F8 000A 01AF  03F0 10426          .WORD   ^X01AF,^X000A,^X03F8,^X03F8,^X03F8,^X03F8,^X0231 ; D2 - D8
03F8 00EE 03F8 03F8 00EE 0390 0271  03FE 10427          .WORD   ^X0271,^X0390,^X00EE,^X03F8,^X03F8,^X00EE,^X03F8 ; D9 - DF
0287 0287 0287 0287 0287 0287 0287  040C 10428          .WORD   ^X0287,^X0287,^X0287,^X0287,^X0287,^X0287,^X0287 ; E0 - E6
024F 026F 01EF 00EF 03F8 03F8 0287  041A 10429          .WORD   ^X0287,^X03F8,^X03F8,^X00EF,^X01EF,^X026F,^X024F ; E7 - ED
03F8 03A1 03A1 028F 0280 020F 022F  0428 10430          .WORD   ^X022F,^X020F,^X0280,^X028F,^X03A1,^X03A1,^X03F8 ; EE - F4
03A5 03A5 031E 02B4 00AE 00AE 03F8  0436 10431          .WORD   ^X03F8,^X00AE,^X00AE,^X02B4,^X031E,^X03A5,^X03A5 ; F5 - FB
          03F8 03F8 0776 0776       0444 10432          .WORD   ^X0776,^X0776,^X03F8,^X03F8                      ; FC - FF
                                    044C 10433 ;
                                    044C 10434 ; Expected CS Addresses for Subtest 3
                                    044C 10435 ;
03F8 03F8 03F8 03F8 03F8 03F8 03F8  044C 10436 31$:     .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 00 - 06
0332 032A 0224 0318 02A8 02A4 03F8  045A 10437          .WORD   ^X03F8,^X02A4,^X02A8,^X0318,^X0224,^X032A,^X0332 ; 07 - 0D
03F8 03F8 03F8 03F8 03F8 02A2 03AA  0468 10438          .WORD   ^X03AA,^X02A2,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 0E - 14
03F8 03F8 03F8 03F8 03F8 03F8 03F8  0476 10439          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 15 - 1B
027C 0294 027C 03F8 03F8 03F8 03F8  0484 10440          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X027C,^X0294,^X027C ; 1C - 22
0302 0302 0294 0368 0294 0360 0294  0492 10441          .WORD   ^X0294,^X0360,^X0294,^X0368,^X0294,^X0302,^X0302 ; 23 - 29
03F8 03B3 03B3 03B1 03B1 0274 0274  04A0 10442          .WORD   ^X0274,^X0274,^X03B1,^X03B1,^X03B3,^X03B3,^X03F8 ; 2A - 30
027C 03BB 0312 0312 00AE 0051 03F8  04AE 10443          .WORD   ^X03F8,^X0051,^X00AE,^X0312,^X0312,^X03BB,^X027C ; 31 - 37
00EE 028F 00EE 030A 030A 0320 02BC  04BC 10444          .WORD   ^X02BC,^X0320,^X030A,^X030A,^X00EE,^X028F,^X00EE ; 38 - 3E
00E6 035D 00E6 035D 00E6 035D 03F8  04CA 10445          .WORD   ^X03F8,^X035D,^X00E6,^X035D,^X00E6,^X035D,^X00E6 ; 3F - 45
00E7 00E7 00E7 00E7 00E7 00E6 035D  04D8 10446          .WORD   ^X035D,^X00E6,^X00E7,^X00E7,^X00E7,^X00E7,^X00E7 ; 46 - 4C
03F8 031D 036C 030D 00E0 00E7 00E7  04E6 10447          .WORD   ^X00E7,^X00E7,^X00E0,^X030D,^X036C,^X031D,^X03F8 ; 4D - 53
03F8 03F8 024E 03F8 01EC 0222 00F4  04F4 10448          .WORD   ^X00F4,^X0222,^X01EC,^X03F8,^X024E,^X03F8,^X03F8 ; 54 - 5A
01DA 035E 02CA 02BA 02B2 02AC 03F8  0502 10449          .WORD   ^X03F8,^X02AC,^X02B2,^X02BA,^X02CA,^X035E,^X01DA ; 5B - 61
00E7 01DA 035E 01DA 035E 01DA 035E  0510 10450          .WORD   ^X035E,^X01DA,^X035E,^X01DA,^X035E,^X01DA,^X00E7 ; 62 - 68
00A0 01DB 01DB 01DB 00E7 00E7 00E7  051E 10451          .WORD   ^X00E7,^X00E7,^X00E7,^X01DB,^X01DB,^X01DB,^X00A0 ; 69 - 6F
036E 022C 00F8 03F8 0315 036C 0315  052C 10452          .WORD   ^X0315,^X036C,^X0315,^X03F8,^X00F8,^X022C,^X036E ; 70 - 76
01E5 03F8 0064 0018 0092 0092 03F8  053A 10453          .WORD   ^X03F8,^X0092,^X0092,^X0018,^X0064,^X03F8,^X01E5 ; 77 - 7D
02A7 006F 025B 004F 023D 03F8 01EE  0548 10454          .WORD   ^X01EE,^X03F8,^X023D,^X004F,^X025B,^X006F,^X02A7 ; 7E - 84
00CF 0291 00AF 028B 008C 0309 001C  0556 10455          .WORD   ^X001C,^X0309,^X008C,^X028B,^X00AF,^X0291,^X00CF ; 85 - 8B
01AF 0052 00EE 02AF 01CF 008F 0279  0564 10456          .WORD   ^X0279,^X008F,^X01CF,^X02AF,^X00EE,^X0052,^X01AF ; 8C - 92
0051 0051 03F8 03F8 03F8 03F8 000A  0572 10457          .WORD   ^X000A,^X03F8,^X03F8,^X03F8,^X03F8,^X0051,^X0051 ; 93 - 99
```

```
 023D 03F8 00EE 028F 000F 00EE 00EE   0580 10458          .WORD   ^X00EE,^X00EE,^X000F,^X028F,^X00EE,^X03F8,^X023D ; 9A - A0
 008C 0309 001C 02A7 006F 025B 004F   058E 10459          .WORD   ^X004F,^X025B,^X006F,^X02A7,^X001C,^X0309,^X008C ; A1 - A7
 01CF 008F 0279 00CF 0291 00AF 028B   059C 10460          .WORD   ^X028B,^X00AF,^X0291,^X00CF,^X0279,^X008F,^X01CF ; A8 - AE
 03F8 03F8 000A 01AF 0052 00EE 02AF   05AA 10461          .WORD   ^X02AF,^X00EE,^X0052,^X01AF,^X000A,^X03F8,^X03F8 ; AF - B5
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   05B8 10462          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; B6 - BC
 006F 025B 004F 023D 03F8 03F8 03F8   05C6 10463          .WORD   ^X03F8,^X03F8,^X03F8,^X023D,^X004F,^X025B,^X006F ; BD - C3
 0291 00AF 028B 008C 0309 001C 02A7   05D4 10464          .WORD   ^X02A7,^X001C,^X0309,^X008C,^X028B,^X00AF,^X0291 ; C4 - CA
 0052 00EE 02AF 01CF 008F 0279 00CF   05E2 10465          .WORD   ^X00CF,^X0279,^X008F,^X01CF,^X02AF,^X00EE,^X0052 ; CB - D1
 0231 03F8 03F8 03F8 03F8 000A 01AF   05F0 10466          .WORD   ^X01AF,^X000A,^X03F8,^X03F8,^X03F8,^X03F8,^X0231 ; D2 - D8
 03F8 00EE 03F8 03F8 00EE 0390 0271   05FE 10467          .WORD   ^X0271,^X0390,^X00EE,^X03F8,^X03F8,^X00EE,^X03F8 ; D9 - DF
 0287 0287 0287 0287 0287 0287 0287   060C 10468          .WORD   ^X0287,^X0287,^X0287,^X0287,^X0287,^X0287,^X0287 ; E0 - E6
 024F 026F 01EF 00EF 03F8 03F8 0287   061A 10469          .WORD   ^X0287,^X03F8,^X03F8,^X00EF,^X01EF,^X026F,^X024F ; E7 - ED
 03F8 03A1 03A1 028F 0280 020F 022F   0628 10470          .WORD   ^X022F,^X020F,^X0280,^X028F,^X03A1,^X03A1,^X03F8 ; EE - F4
 03A5 03A5 031E 02B4 00AE 00AE 03F8   0636 10471          .WORD   ^X03F8,^X00AE,^X00AE,^X02B4,^X031E,^X03A5,^X03A5 ; F5 - FB
                03F8 03F8 0776 0776   0644 10472          .WORD   ^X0776,^X0776,^X03F8,^X03F8                      ; FC - FF
                                      064C 10473
                                      064C 10474 BEGIN_CPU_DATA
                                      0002 10475
                                      0002 10476 DCS_DATA
                                      0001 10477
U 00, 7700,C800,0364,0300,0470,1801   0001 10478 ;% 00   NOP                                  ; get cpu up to speed
U 01, 5700,C840,0364,0308,0470,1802   000C 10482 ;% 01   R[TEMP2]_RDM                          ; load opcode in R[TEMP2]
U 02, 7700,F800,4ECF,4FFF,8470,1803   0017 10486 ;% 02   LONLIT_[62616000]                    ; load opspecs in lonlit reg.
U 03, 7700,8800,5BE5,03D4,0470,1804   0022 10490 ;% 03   Q_R[LONLIT]                          ; move opspecs to Q register
U 04, 7700,0842,00A4,0308,0470,1805   002D 10494 ;% 04   R[TEMP2]_M[TEMP2].OR.Q               ; OR opcode with opspecs
U 05, 7700,4868,5BE4,0308,0470,1806   0038 10498 ;% 05   M[TEMP8]_R[TEMP2]                    ; save in M[TEMP8] for error
U 06, 7700,C800,5BE4,03D8,4A70,1807   0043 10502 ;% 06   VA_R[ZERO]                          ; zero the VA register
U 07, 7701,C800,5BE4,0208,5590,1808   004E 10506 ;% 07   WRITE.LONG R[TEMP2]                 ; write data for XB0
U 08, 7700,0800,5BE4,0308,5470,1809   0059 10510 ;% 08   WDR_UNROT(R[TEMP2])                 ; validate cache location 0
U 09, 7700,C800,0364,0300,4470,180A   0064 10514 ;% 09   VA_VA+4                             ; update the VA by 4
U 0A, 7700,3800,4CCD,4DCE,0470,180B   006F 10518 ;% 0A   LONLIT_[66656463]                   ; load opspecs in lonlit reg.
U 0B, 7700,4869,5BE4,03D4,0470,180C   007A 10522 ;% 0B   M[TEMP9]_R[LONLIT]                  ; save in M[TEMP9] for error
U 0C, 7701,8800,5BE4,02D4,5590,180D   0085 10526 ;% 0C   WRITE.LONG R[LONLIT]               ; write data for XB1
U 0D, 7700,C800,5BE4,03D4,5470,180E   0090 10530 ;% 0D   WDR_UNROT(R[LONLIT])               ; validate cache location 4
U 0E, 7700,C800,5BE4,03D8,4870,180F   009B 10534 ;% 0E   PC_R[ZERO]                          ; zero PC to force prefetch
U 0F, 7700,C800,0364,1700,0470,1810   00A6 10538 ;% 0F   BUT/IRD1TST                         ; perform ird1
U 10, 7300,4800,0364,0300,0470,1811   00B1 10542 ;% 10   NOP,STOP.CLOCK                      ; halt
                                      00BC 10546
U 11, 7700,4800,0364,0300,0470,1812   00BC 10547 ;% 11   NOP                                  ; get cpu up to speed
U 12, 7700,4800,0364,1B00,0470,1800   00C7 10551 ;% 12   BUT/LOD.INC.BRA,NEXT/1800           ; load OSR and incr. IRD cnter
U 13, 7300,4800,0364,0300,0470,1814   00D2 10555 ;% 13   STOP.CLOCK                          ; halt
                                      00DD 10559
U 14, 7700,C800,0364,0300,0470,1815   00DD 10560 ;% 14   NOP                                  ; get cpu up to speed
U 15, 5700,C800,0364,0300,2A70,1816   00E8 10564 ;% 15   WCTRL/FPA.ENABLE_WB5,WB_RDM          ; write the FPA enable bit
U 16, 7700,4800,0364,0300,0470,1817   00F3 10568 ;% 16   NOP                                  ; ...
U 17, 7700,4800,0364,C300,0470,1800   00FE 10572 ;% 17   BUT/NO.FPA,NEXT/1800               ; check FPA status
U 18, 7300,C800,0364,0300,0470,1819   0109 10576 ;% 18   STOP.CLOCK                          ; halt
                                      0114 10580
U 19, 7700,C800,0364,0300,0470,181A   0114 10581 ;% 19   NOP                                  ; get the cpu up to speed
U 1A, 7700,F800,7F9F,FFFF,8470,181B   011F 10585 ;% 1A   LONLIT_[00C00000]                   ; address of s&c bit in crar
U 1B, 7700,4800,5BE4,03D4,2470,181C   012A 10589 ;% 1B   WCTRL/LOADCRAR,WB_R[LONLIT]         ; address the s&c bit
U 1C, 7700,B800,7FDF,FFFF,8470,181D   0135 10593 ;% 1C   LONLIT_[00400000]                   ; data for ctrl-p int. disable
U 1D, 7700,4800,5BE4,03D4,2C70,181E   0140 10597 ;% 1D   WCTRL/CONWRITE,WB_R[LONLIT]        ; disable ctrl-p interrupts
U 1E, 7300,C800,0364,0300,0470,181F   014B 10601 ;% 1E   STOP.CLOCK                          ; halt
                                      0156 10605
```

```
U 1F. 7700,C800,0364,0300,0470,1820  0156 10606 ;* 1F   NOP                                    ; get the cpu up to speed
U 20. 7700,F800,7F9F,FFFF,8470,1821  0161 10610 ;* 20   LONLIT_[00C00000]                       ; address of s&c bit in crar
U 21. 7700,C800,5BE4,03D4,2470,1822  016C 10614 ;* 21   WCTRL/LOADCRAR,WB_R[LONLIT]            ; address the s&c bit
U 22. 7700,7800,7FB7,FFFF,8470,1823  0177 10618 ;* 22   LONLIT_[00900000]                       ; data for ctrl-p int. enable
U 23. 7700,4800,5BE4,03D4,2C70,1824  0182 10622 ;* 23   WCTRL/CONWRITE,WB_R[LONLIT]            ; enable ctrl-p interrupts
U 24. 7300,C800,0364,0300,0470,1825  018D 10626 ;* 24   STOP.CLOCK                            ; halt
                                     0198 10630
                                     0198 10631 ; Set-up RTEMP0 and RTEMP1 for error callout
                                     0198 10632
U 25. 7700,C800,0364,0300,0470,1826  0198 10633 ;* 25   NOP                                    ; get cpu up to speed
U 26. 7700,0848,5924,0300,0470,1827  01A3 10637 ;* 26   R[TEMP0]_M[TEMP8]                      ; move XB0 data to R[TEMP0]
U 27. 7700,0849,5924,0304,0470,1828  01AE 10641 ;* 27   R[TEMP1]_M[TEMP9]                      ; move XB1 data to R[TEMP1]
U 28. 7300,C800,0364,0300,0470,1829  01B9 10645 ;* 28   STOP.CLOCK                            ; halt
                                     01C4 10649
                                     01C4 10650 RTEMP_DATA
                                     0001 10651
                          00000000   0001 10652          .LONG   0                    ; rtemp0 = contents of xb0
                          00000000   0005 10653          .LONG   0                    ; rtemp1 = contents of xb1
                          00000000   0009 10654          .LONG   0                    ; rtemp2 = instruction
                                     000D 10655
                                     000D 10656 END_CPU_DATA
                                     064C 10657
                                     064C 10658 END_TEST_DATA
                                     064C
                                     064C        ;-------------------------------------------------------------------------
                                     064C 10659
                                     064C 10660 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
                        0032       .SBTTL  "TEST   0CB     Second IRDX Instruction Address Decode Test 2
                        0032 10662 ;*******************************************************************************
                        0032 10663 ;++
                        0032 10664 ;
                        0032 10665 ; FUNCTIONAL DESCRIPTION
                        0032 10666 ;
                        0032 10667 ;       This test will examine the signal ROM OS INH H from the Native Mode
                        0032 10668 ;       IRDX roms when the IRD Counter is equal to one and the operand
                        0032 10669 ;       specifier is for a memory reference.
                        0032 10670 ;
                        0032 10671 ; TEST ALGORITHM
                        0032 10672 ;
                        0032 10673 ;       This test consists of three subtests.  The first subtest will examine
                        0032 10674 ;       the IRDX roms when the Floating Point Accelerator (FPA) is not present
                        0032 10675 ;       in the system.  The second subtest will examine the IRDX roms when the
                        0032 10676 ;       FPA is present but disabled.  The third subtest will examine the IRDX
                        0032 10677 ;       roms when the FPA is present and enabled.  The algorithm is as follows:
                        0032 10678 ;
                        0032 10679 ;        1.  Initialize the processor.
                        0032 10680 ;        2.  Disable CTRL/P interrupts.
                        0032 10681 ;        3.  Check to see if FPA present
                        0032 10682 ;               If FPA present
                        0032 10683 ;                   Then skip to Subtest 2
                        0032 10684 ;               Else fall through to subtest 1
                        0032 10685 ;
                        0032 10686 ;       Subtest 1
                        0032 10687 ;
                        0032 10688 ;        4.  Set-up flag to expect a micro trap.
                        0032 10689 ;        5.  Initialize loop I for 256 opcodes (00-FF).
                        0032 10690 ;        6.  Store the index I in a location that will be used to load into the
                        0032 10691 ;            WDREG.  The value is used as the opcode.
                        0032 10692 ;        7.  Loop on error from here.
                        0032 10693 ;        8.  Set-up DCS location 12 with a BUT/LOD.INC.BRA?,NEXT/1800.  This
                        0032 10694 ;            is used to get the IRD counter to 1 without sourcing an address
                        0032 10695 ;            from the IRDX roms (ie., by doing an IRDX).
                        0032 10696 ;        9.  Execute the following test microcode (performs the IRD1):
                        0032 10697 ;               a.  NOP to get the RDM up to speed.
                        0032 10698 ;               b.  Read the WDREG into RTEMP2 (opcode).
                        0032 10699 ;               c.  Load the operands into the LONLIT register.
                        0032 10700 ;               d.  Store the operands into the Q register to prepare for ORing
                        0032 10701 ;                   the opcode with the operands.
                        0032 10702 ;               e.  OR the opcode and operands together (XB0 data).
                        0032 10703 ;               f.  Store XB0 data in M[TEMP8] in case needed for error callout.
                        0032 10704 ;               g.  Zero the VA.
                        0032 10705 ;               h.  Validate cache location 0 by first writing the data in
                        0032 10706 ;                   RTEMP2 then reading the data back.
                        0032 10707 ;               i.  Update the VA by 4.
                        0032 10708 ;               j.  Load the second operand data into the LONLIT register
                        0032 10709 ;                   (XB1 data).
                        0032 10710 ;               k.  Store XB1 data in M[TEMP9] in case needed for error callout.
                        0032 10711 ;               l.  Validate cache location 4 by first writing the data in
                        0032 10712 ;                   the LONLIT register then reading the data back.
                        0032 10713 ;               m.  Load the PC with zero to cause a prefetch to take place.
                        0032 10714 ;               n.  Do an IRD1 to init the IRD counter and prepare for instruc-
                        0032 10715 ;                   tion/operand specifier evaluation.
```

```
0032 10716 ;                    o.  Halt.
0032 10717 ;            10.  Perform the following test microcode (performs the LOD.INC.BRA?)
0032 10718 ;                    a.  NOP to get the RDM up to speed.
0032 10719 ;                    b.  Do an BUT/LOD.INC.BRA?,NEXT/1800
0032 10720 ;                    c.  Halt.
0032 10721 ;            11.  Load DCS location 12 with a BUT/IRDX micro order.
0032 10722 ;            12.  Perform the following test microcode (performs the IRDX with the
0032 10723 ;                 IRD counter = 1):
0032 10724 ;                    a.  NOP to get the RDM up to speed.
0032 10725 ;                    b.  Do an IRDX and strobe the VBUS.
0032 10726 ;                    c.  Halt.
0032 10727 ;            13.  Compare the VBUS data with the expected value and bit position
0032 10728 ;                 from the VBUS Table.
0032 10729 ;            14.  If the two do not equal, execute the following microcode and
0032 10730 ;                 signal an error (this is done to set-up for printing RTEMP0 & 1
0032 10731 ;                 on the error callout):
0032 10732 ;                    a.  NOP to get the RDM up to speed.
0032 10733 ;                    b.  Load RTEMP0 with the contents of M[TEMP8]
0032 10734 ;                    c.  Load RTEMP1 with the contents of M[TEMP9]
0032 10735 ;                    d.  Halt
0032 10736 ;            15.  Continue with loop I for the 256 VAX opcodes.
0032 10737 ;            16.  Re-enable CTRL/P interrupts then skip to next tests.
0032 10738 ;
0032 10739 ;            Subtest 2
0032 10740 ;
0032 10741 ;            17.  Load the FPA disable bit (bit <5> equal to 0).
0032 10742 ;            18.  Loop on error from here.
0032 10743 ;            19.  Set-up to expect micro trap.
0032 10744 ;            20.  Execute the test microcode.  This is done only to get the FPA to
0032 10745 ;                 a state where it can be disabled or enabled.  If this had not been
0032 10746 ;                 done before then the FPA would be setting in the initialize state
0032 10747 ;                 which forces FPA enable.
0032 10748 ;            21.  Execute the following microcode to disable the FPA:
0032 10749 ;                    a.  NOP to get the RDM up to speed.
0032 10750 ;                    b.  Read the RDM WDREG onto the W Bus and then write the W Bus
0032 10751 ;                        to the FPA.
0032 10752 ;                    c.  Perform a NO.FPA test to insure that the FPA got disabled.
0032 10753 ;                    d.  Halt.
0032 10754 ;            22.  Check the Trap register to insure that the FPA got disabled.
0032 10755 ;            23.  If the FPA is not disabled, then signal an error.
0032 10756 ;            24.  Set-up flag to expect a micro trap.
0032 10757 ;            25.  Initialize loop I for 256 opcodes (00-FF).
0032 10758 ;            26.  Store the index I in a location that will be used to load into the
0032 10759 ;                 WDREG.  The value is used as the opcode.
0032 10760 ;            27.  Loop on error from here.
0032 10761 ;            28.  Set-up DCS location 12 with a BUT/LOD.INC.BRA?,NEXT/1800.  This
0032 10762 ;                 is used to get the IRD counter to 1 without sourcing an address
0032 10763 ;                 from the IRDX roms (ie., by doing an IRDX).
0032 10764 ;            29.  Execute the following test microcode:
0032 10765 ;                    a.  NOP to get the RDM up to speed.
0032 10766 ;                    b.  Read the WDREG into RTEMP2 (opcode).
0032 10767 ;                    c.  Load the operands into the LONLIT register.
0032 10768 ;                    d.  Store the operands into the Q register to prepare for ORing
0032 10769 ;                        the opcode with the operands.
0032 10770 ;                    e.  OR the opcode and operands together (XB0 data).
```

```
0032 10771 ;                f.  Store XB0 data in M[TEMP8] in case needed for error callout.
0032 10772 ;                g.  Zero the VA.
0032 10773 ;                h.  Validate cache location 0 by first writing the data in
0032 10774 ;                    RTEMP2 then reading the data back.
0032 10775 ;                i.  Update the VA by 4.
0032 10776 ;                j.  Load the second operand data into the LONLIT register
0032 10777 ;                    (XB1 data).
0032 10778 ;                k.  Store XB1 data in M[TEMP9] in case needed for error callout.
0032 10779 ;                l.  Validate cache location 4 by first writing the data in
0032 10780 ;                    the LONLIT register then reading the data back.
0032 10781 ;                m.  Load the PC with zero to cause a prefetch to take place.
0032 10782 ;                n.  Do an IRD1 to init the IRD counter and prepare for instruc-
0032 10783 ;                    tion/operand specifier evaluation.
0032 10784 ;                o.  Halt.
0032 10785 ;        30.  Perform the following test microcode.
0032 10786 ;                a.  NOP to get the RDM up to speed.
0032 10787 ;                b.  Do an BUT/LOD.INC.BRA?,NEXT/1800
0032 10788 ;                c.  Halt.
0032 10789 ;        31.  Load DCS location 12 with a BUT/IRDX micro order.
0032 10790 ;        32.  Perform the following test microcode (performs the IRDX with the
0032 10791 ;             IRD counter = 1):
0032 10792 ;                a.  NOP to get the RDM up to speed.
0032 10793 ;                b.  Do an IRDX and strobe the VBUS.
0032 10794 ;                c.  Halt.
0032 10795 ;        33.  Compare the VBUS data with the expected value and bit position
0032 10796 ;             from the VBUS Table.
0032 10797 ;        34.  If the two do not equal, execute the following microcode and
0032 10798 ;             signal an error (this is done to set-up for printing RTEMP0 & 1
0032 10799 ;             on the error callout):
0032 10800 ;                a.  NOP to get the RDM up to speed.
0032 10801 ;                b.  Load RTEMP0 with the contents of M[TEMP8]
0032 10802 ;                c.  Load RTEMP1 with the contents of M[TEMP9]
0032 10803 ;                d.  Halt
0032 10804 ;        35.  Continue with loop I for the 256 VAX opcodes.
0032 10805 ;        36.  Go to Subtest 3.
0032 10806 ;
0032 10807 ;        Subtest 3
0032 10808 ;
0032 10809 ;        37.  Load the FPA enable bit (bit <5> equal to 1).
0032 10810 ;        38.  Loop on error from here.
0032 10811 ;        39.  Set-up to expect micro trap.
0032 10812 ;        40.  Execute the following microcode to enable the FPA:
0032 10813 ;                a.  NOP to get the RDM up to speed.
0032 10814 ;                b.  Read the RDM WDREG onto the W Bus and then write the W Bus
0032 10815 ;                    to the FPA.
0032 10816 ;                c.  Perform a NO.FPA test to insure that the FPA got enabled.
0032 10817 ;                d.  Halt.
0032 10818 ;        41.  Check the Trap register to insure that the FPA got enabled.
0032 10819 ;        42.  If the FPA is not enabled, then signal an error.
0032 10820 ;        43.  Set-up flag to expect a micro trap.
0032 10821 ;        44.  Initialize loop I for 256 opcodes (00-FF).
0032 10822 ;        45.  Store the index I in a location that will be used to load into the
0032 10823 ;             WDREG.  The value is used as the opcode.
0032 10824 ;        46.  Loop on error from here.
0032 10825 ;        47.  Set then Clear the DCLO bit in the RDCTRL register to perform an
```

```
0032 10826 ;            initialize.  This is done because when the test microcode
0032 10827 ;            executes, if the opcode was an FP opcode then there is a
0032 10828 ;            possibility that the processor could be setting there in an
0032 10829 ;            FPAWAIT state (M clock and D clock stalled).  If this were to
0032 10830 ;            happen then the cpu clocks would no longer run and the remaining
0032 10831 ;            test opcodes would fail.
0032 10832 ;       48.  Set-up DCS location 12 with a BUT/LOD.INC.BRA?,NEXT/1800.  This
0032 10833 ;            is used to get the IRD counter to 1 without sourcing an address
0032 10834 ;            from the IRDX roms (ie., by doing an IRDX).
0032 10835 ;       49.  Execute the following test microcode:
0032 10836 ;               a.  NOP to get the RDM up to speed.
0032 10837 ;               b.  Read the WDREG into RTEMP2 (opcode).
0032 10838 ;               c.  Load the operands into the LONLIT register.
0032 10839 ;               d.  Store the operands into the Q register to prepare for ORing
0032 10840 ;                   the opcode with the operands.
0032 10841 ;               e.  OR the opcode and operands together (XB0 data).
0032 10842 ;               f.  Store XB0 data in M[TEMP8] in case needed for error callout.
0032 10843 ;               g.  Zero the VA.
0032 10844 ;               h.  Validate cache location 0 by first writing the data in
0032 10845 ;                   RTEMP2 then reading the data back.
0032 10846 ;               i.  Update the VA by 4.
0032 10847 ;               j.  Load the second operand data into the LONLIT register
0032 10848 ;                   (XB1 data).
0032 10849 ;               k.  Store XB1 data in M[TEMP9] in case needed for error callout.
0032 10850 ;               l.  Validate cache location 4 by first writing the data in
0032 10851 ;                   the LONLIT register then reading the data back.
0032 10852 ;               m.  Load the PC with zero to cause a prefetch to take place.
0032 10853 ;               n.  Do an IRD1 to init the IRD counter and prepare for instruc-
0032 10854 ;                   tion/operand specifier evaluation.
0032 10855 ;               o.  Halt.
0032 10856 ;       50.  Perform the following test microcode.
0032 10857 ;               a.  NOP to get the RDM up to speed.
0032 10858 ;               b.  Do an BUT/LOD.INC.BRA?,NEXT/1800
0032 10859 ;               c.  Halt.
0032 10860 ;       51.  Load DCS location 12 with a BUT/IRDX micro order.
0032 10861 ;       52.  Perform the following test microcode (performs the IRDX with the
0032 10862 ;            IRD counter = 1):
0032 10863 ;               a.  NOP to get the RDM up to speed.
0032 10864 ;               b.  Do an IRDX and strobe the VBUS.
0032 10865 ;               c.  Halt.
0032 10866 ;       53.  Compare the VBUS data with the expected value and bit position
0032 10867 ;            from the VBUS Table.
0032 10868 ;       54.  If the two do not equal, execute the following microcode and
0032 10869 ;            signal an error (this is done to set-up for printing RTEMP0 & 1
0032 10870 ;            on the error callout):
0032 10871 ;               a.  NOP to get the RDM up to speed.
0032 10872 ;               b.  Load RTEMP0 with the contents of M[TEMP8]
0032 10873 ;               c.  Load RTEMP1 with the contents of M[TEMP9]
0032 10874 ;               d.  Halt
0032 10875 ;       55.  Continue with loop I for the 256 VAX opcodes.
0032 10876 ;       56.  Re-enable CTRL/P interrupts.
0032 10877 ;       57.  Skip to next tests.
0032 10878 ;
0032 10879 ; TEST PATTERNS
0032 10880 ;
```

```
0032 10881 ;       For this test the opcodes are generated by a simple count pattern.  The
0032 10882 ;       opcodes are based on the 256 base machine instructions as specified in
0032 10883 ;       the VAX architecture handbook.  The generated opcodes are written to
0032 10884 ;       the CPU and combined (ORed) with three bytes of operand data to form
0032 10885 ;       a machine instruction.  The operands all specify Register Deferred mode
0032 10886 ;       using registers 0-6 and are loaded from the LONLIT register.
0032 10887 ;
0032 10888 ;       Also required are two tables of expected VBUS values for the second
0032 10889 ;       IRDX cycle.  One table contains the VBUS values for the second IRDX
0032 10890 ;       when the FPA is disabled, the other for when it is enabled.
0032 10891 ;
0032 10892 ; LOGIC DESCRIPTION
0032 10893 ;
0032 10894 ;       This test will check the validity of the IRDX roms on the DPM module.
0032 10895 ;       The IRDX roms are addressed by bits <7:0> of the Instruction Register.
0032 10896 ;       Along with being addressed by the IR, there are three other signals
0032 10897 ;       that affect the address.  They are IRD CTR 0 H, FPA ENABLED L, and
0032 10898 ;       REG MODE H.  This test will examine every 8th(hex) address starting at
0032 10899 ;       address 6 with FPA disabled and starting at address 4 with FPA enabled.
0032 10900 ;
0032 10901 ; ASSUMPTIONS
0032 10902 ;
0032 10903 ;       This test assumes the following logic has already been tested; the
0032 10904 ;       CPU's data path, the cache, the XB, and the IR and OSR.
0032 10905 ;
0032 10906 ; ERROR DESCRIPTION
0032 10907 ;
0032 10908 ;       Subtest 1
0032 10909 ;
0032 10910 ;               Expected CS Address
0032 10911 ;               Received CS Address
0032 10912 ;
0032 10913 ;       Subtest 2
0032 10914 ;
0032 10915 ;               First IFERROR
0032 10916 ;
0032 10917 ;                       Expected CS Address
0032 10918 ;                       Received CS Address
0032 10919 ;
0032 10920 ;               Second IFERROR
0032 10921 ;
0032 10922 ;                       Expected VBUS ( 00000ynn )      y  = VBUS bit value
0032 10923 ;                       Received VBUS ( 00000ynn )      nn = VBUS bit number
0032 10924 ;                       Loop I count
0032 10925 ;                       Contents of XB0
0032 10926 ;                       Contents of XB1
0032 10927 ;
0032 10928 ;       Subtest 3
0032 10929 ;
0032 10930 ;               First IFERROR
0032 10931 ;
0032 10932 ;                       Expected CS Address
0032 10933 ;                       Received CS Address
0032 10934 ;
0032 10935 ;               Second IFERROR
```

```
                        0032 10936 ;
                        0032 10937 ;                       Expected VBUS ( 00000ynn )      y  = VBUS bit value
                        0032 10938 ;                       Received VBUS ( 00000ynn )      nn = VBUS bit number
                        0032 10939 ;                       Loop I count
                        0032 10940 ;                       Contents of XB0
                        0032 10941 ;                       Contents of XB1
                        0032 10942 ;--
                        0032 10943 ;*****************************************************************************
                        0032 10944
                        0032 10945          INITIALIZE                                  ; initialize the comet cpu
                        0034 10946          FETCH           <^X19>                      ; disable ctrl-p interrupts
                        0039 10947          BRSTCLK         <^X1E>                      ; ...
                        003D 10948          FETCH           <^X16>                      ; execute a but/no.fpa to check
                        0042 10949          BRSTCLK         <^X18>                      ; for fpa present
                        0046 10950          CMPREGMSK       CSTRP,1$,,2$,,W             ; is the fpa present?
                        0052 10951          SKIP            100$,ONERROR                ; if so then skip to subtest 2
                        0059 10952
                        0059 10953          SUBTEST         1
                        005C
                        005C            ;////////////////////////////////////////////////////////////////////////////
                        005C 10954 ;*
                        005C 10955 ;
                        005C 10956 ; FPA is not present.
                        005C 10957 ;
                        005C 10958 ;-
                        005C 10959          EXPUTRAP                                    ; else expect non-dcs addresses
                        005E 10960          LOOP            I,1,256                     ; loop for 256 opcodes
                        0065 10961          SAVEINDEX       I,9$                        ; save the loop count
                        006A 10962          LOADREG         WDREG,9$,,L                 ; load wdreg with loop count
                        0072 10963          ERRLOOP                                     ; loop on error from here
                        0074 10964          LOADDCS         20$,,<^X12>,1               ; load BUT/LOD.INC.BRA?
                        007B 10965          FETCH           0                           ; execute ird1tst
                        0080 10966          BRSTCLK         <^X10>,INH                  ; ...
                        0084 10967          FETCH           <^X11>                      ; execute BUT/LOD.INC.BRA?
                        0089 10968          BRSTCLK         <^X13>,INH                  ; ...
                        008D 10969          LOADDCS         21$,,<^X12>,1               ; load BUT/IRDX
                        0094 10970          FETCH           <^X11>                      ; execute the second irdx
                        0099 10971          BRSTCLK         <^X12>,INH                  ; ...
                        009D 10972          CMPVBUS         30$,I                       ; check VBUS bit for an error
                        00A2 10973          SKIP            50$,NOERROR                 ; continue testing if no error
                        00A9 10974          FETCH           <^X25>                      ; start dcs program
                        00AE 10975          BRSTCLK         <^X28>                      ; end dcs program
                        00B2 10976          IFERROR         2,<<IRD>>,<<MIC>>           ; signal an error
                        00B9 10977 50$:     ENDLOOP         I                           ; continue loop
                        00BC 10978          FETCH           <^X1F>                      ; re-enable ctrl-p interrupts
                        00C1 10979          BRSTCLK         <^X24>                      ; ...
                        00C5 10980          SKIP                                        ; skip on to the next test
                        00CC 10981
                        00CC 10982 100$:    SUBTEST         2
                        00CF
                        00CF            ;////////////////////////////////////////////////////////////////////////////
                        00CF 10983 ;*
                        00CF 10984 ;
                        00CF 10985 ; FPA present but not enabled
                        00CF 10986 ;
```

```
00CF 10987 ;-
00CF 10988          LOADREG         WDREG,7$,,L             ; load fpa disable bit
00D7 10989          ERRLOOP                                 ; loop on error here
00D9 10990          EXPUTRAP                                ; expect micro trap
00DB 10991          FETCH           0                       ; start dcs program
00E0 10992          BRSTCLK         <^X10>,INH              ; end dcs program
00E4 10993          FETCH           <^X14>                  ; execute the disable
00E9 10994          BRSTCLK         <^X18>,INH              ; and read back fpa status
00ED 10995          CMPREGMSK       CSTRP,3$,,5$,,W         ; is the fpa disabled?
00F9 10996          IFERROR         ,,<FPA>                 ; if not, signal an error
00FF 10997          EXPUTRAP                                ; else expect non-dcs addresses
0101 10998          LOOP            I,1,256                 ; loop for 256 opcodes
0108 10999          SAVEINDEX       I,9$                    ; save the loop count
010D 11000          LOADREG         WDREG,9$,,L             ; load wdreg with loop count
0115 11001          ERRLOOP                                 ; loop on error from here
0117 11002          LOADDCS         20$,,<^X12>,1           ; load BUT/LOD.INC.BRA?
011E 11003          FETCH           0                       ; execute ird1tst
0123 11004          BRSTCLK         <^X10>,INH              ; ...
0127 11005          FETCH           <^X11>                  ; execute BUT/LOD.INC.BRA?
012C 11006          BRSTCLK         <^X13>,INH              ; ...
0130 11007          LOADDCS         21$,,<^X12>,1           ; load BUT/IRDX
0137 11008          FETCH           <^X11>                  ; execute the second irdx
013C 11009          BRSTCLK         <^X13>,INH              ; ...
0140 11010          CMPVBUS         30$,I                   ; check VBUS bit for an error
0145 11011          SKIP            51$,NOERROR             ; continue testing if no error
014C 11012          FETCH           <^X25>                  ; start dcs program
0151 11013          BRSTCLK         <^X28>                  ; end dcs program
0155 11014          IFERROR         2,<<IRD>>,<<MIC>>       ; signal an error
015C 11015 51$:     ENDLOOP         I                       ; continue loop
015F 11016
015F 11017          SUBTEST         3
0162
0162        ;////////////////////////////////////////////////////////////////////////////////
0162 11018 ;+
0162 11019 ;
0162 11020 ; FPA present and enabled
0162 11021 ;
0162 11022 ;-
0162 11023          LOADREG         WDREG,8$,,L             ; load fpa enable bit
016A 11024          ERRLOOP                                 ; loop on error from here
016C 11025          FETCH           <^X14>                  ; enable the fpa
0171 11026          BRSTCLK         <^X18>,INH              ; and read back fpa enable
0175 11027          CMPREGMSK       CSTRP,4$,,5$,,W         ; is the fpa enabled?
0181 11028          IFERROR         ,,<FPA>                 ; if not, signal an error
0187 11029          EXPUTRAP                                ; else expect non-dcs addresses
0189 11030          LOOP            I,1,256                 ; loop for 256 opcodes
0190 11031          SAVEINDEX       I,9$                    ; save the loop count
0195 11032          LOADREG         WDREG,9$,,L             ; load wdreg with loop count
019D 11033          ERRLOOP                                 ; loop on error from here
019F 11034          LOADDCS         20$,,<^X12>,1           ; load BUT/LOD.INC.BRA?
01A6 11035          LOADREG         RDCTRL,10$,,B           ; assert dclo
01AE 11036          LOADREG         RDCTRL,11$,,B           ; de-assert dclo
01B6 11037          FETCH           0                       ; execute ird1tst
01BB 11038          BRSTCLK         <^X10>,INH              ; ...
01BF 11039          FETCH           <^X11>                  ; execute BUT/LOD.INC.BRA?
```

```
                                01C4 11040          BRSTCLK         <^X13>,INH                  ; ...
                                01C8 11041          LOADDCS         21$,,<^X12>,1               ; load BUT/IRDX
                                01CF 11042          FETCH           <^X11>                      ; execute the second irdx
                                01D4 11043          BRSTCLK         <^X13>,INH                  ; ...
                                01D8 11044          CMPVBUS         31$,I                       ; check VBUS bit for an error
                                01DD 11045          SKIP            52$,NOERROR                 ; continue testing if no error
                                01E4 11046          FETCH           <^X25>                      ; start dcs program
                                01E9 11047          BRSTCLK         <^X28>                      ; end dcs program
                                01ED 11048          IFERROR         2,<<IRD>>,<<MIC>>           ; signal an error
                                01F4 11049 52$:     ENDLOOP         I                           ; continue loop
                                01F7 11050          FETCH           <^X1F>                      ; re-enable ctrl-p interrupts
                                01FC 11051          BRSTCLK         <^X24>                      ; ...
                                0200 11052          SKIP                                        ; end of test
                                0207 11053
                                0207 11054 BEGIN_TEST_DATA
                                0207
                                0207       ;-----------------------------------------------------------------------
                                0207 11055
                          1802  0207 11056 1$:      .WORD   ^X1802
                          3FFE  0209 11057 2$:      .WORD   ^X3FFE
                                020B 11058
                          1801  020B 11059 3$:      .WORD   ^X1801
                          1800  020D 11060 4$:      .WORD   ^X1800
                          3FFD  020F 11061 5$:      .WORD   ^X3FFD
                                0211 11062
                          3FFF  0211 11063 6$:      .WORD   ^X3FFF
                                0213 11064
                      00000000  0213 11065 7$:      .LONG   0
                      00000020  0217 11066 8$:      .LONG   ^X00000020
                                021B 11067
                      00000000  021B 11068 9$:      .LONG   0
                            40  021F 11069 10$:     .BYTE   DC_LOW
                            00  0220 11070 11$:     .BYTE   ^X00
                                0221 11071
                                0221 11072
                                0221 11073 ; Loadable microcode
                                0221 11074
                                0221 11075 20$:
U 12, 7700,4800,0364,1B00,0470,1800  0221 11076 ;% 12   BUT/LOD.INC.BRA,NEXT/1800                  ; load OSR & increment IRD cntr
                                022C 11080 21$:
U 12, 7600,8800,0364,0700,0470,1813  022C 11081 ;% 12   STROBE.V.BUS,BUT/IRDX                      ; do irdx and strobe VBUS
                                0237 11085
                                0237 11086 ;
                                0237 11087 ; Expected VBUS data for Subtests 1 and 2
                                0237 11088 ;
                                0237 11089 ; The following is a table of all the expected VBUS values for the second IRDX.
                                0237 11090 ; A 1 means the ROM OS INH H signal is asserted and a 0 means it is not
                                0237 11091 ; asserted.
                                0237 11092 ;
                                0237 11093
                            01  0237 11094 30$:     .BYTE   ^X01
                                0238 11095          VBUSDATA        <^X1E>,1                    ;00
                            01  0239 11096          .BYTE   ^X01
                                023A 11097          VBUSDATA        <^X1E>,1                    ;01
                            01  023B 11098          .BYTE   ^X01
```

```
                023C 11099          VBUSDATA        <^X1E>,1                ;02
             01 023D 11100          .BYTE   ^X01
                023E 11101          VBUSDATA        <^X1E>,1                ;03
             01 023F 11102          .BYTE   ^X01
                0240 11103          VBUSDATA        <^X1E>,1                ;04
             01 0241 11104          .BYTE   ^X01
                0242 11105          VBUSDATA        <^X1E>,1                ;05
             01 0243 11106          .BYTE   ^X01
                0244 11107          VBUSDATA        <^X1E>,1                ;06
             01 0245 11108          .BYTE   ^X01
                0246 11109          VBUSDATA        <^X1E>,1                ;07
             01 0247 11110          .BYTE   ^X01
                0248 11111          VBUSDATA        <^X1E>,1                ;08
             01 0249 11112          .BYTE   ^X01
                024A 11113          VBUSDATA        <^X1E>,1                ;09
             01 024B 11114          .BYTE   ^X01
                024C 11115          VBUSDATA        <^X1E>,1                ;0A
             01 024D 11116          .BYTE   ^X01
                024E 11117          VBUSDATA        <^X1E>,1                ;0B
             01 024F 11118          .BYTE   ^X01
                0250 11119          VBUSDATA        <^X1E>,1                ;0C
             01 0251 11120          .BYTE   ^X01
                0252 11121          VBUSDATA        <^X1E>,1                ;0D
             01 0253 11122          .BYTE   ^X01
                0254 11123          VBUSDATA        <^X1E>,1                ;0E
             01 0255 11124          .BYTE   ^X01
                0256 11125          VBUSDATA        <^X1E>,1                ;0F
             01 0257 11126          .BYTE   ^X01
                0258 11127          VBUSDATA        <^X1E>,1                ;10
             01 0259 11128          .BYTE   ^X01
                025A 11129          VBUSDATA        <^X1E>,1                ;11
             01 025B 11130          .BYTE   ^X01
                025C 11131          VBUSDATA        <^X1E>,1                ;12
             01 025D 11132          .BYTE   ^X01
                025E 11133          VBUSDATA        <^X1E>,1                ;13
             01 025F 11134          .BYTE   ^X01
                0260 11135          VBUSDATA        <^X1E>,1                ;14
             01 0261 11136          .BYTE   ^X01
                0262 11137          VBUSDATA        <^X1E>,1                ;15
             01 0263 11138          .BYTE   ^X01
                0264 11139          VBUSDATA        <^X1E>,1                ;16
             01 0265 11140          .BYTE   ^X01
                0266 11141          VBUSDATA        <^X1E>,1                ;17
             01 0267 11142          .BYTE   ^X01
                0268 11143          VBUSDATA        <^X1E>,1                ;18
             01 0269 11144          .BYTE   ^X01
                026A 11145          VBUSDATA        <^X1E>,1                ;19
             01 026B 11146          .BYTE   ^X01
                026C 11147          VBUSDATA        <^X1E>,1                ;1A
             01 026D 11148          .BYTE   ^X01
                026E 11149          VBUSDATA        <^X1E>,1                ;1B
             01 026F 11150          .BYTE   ^X01
                0270 11151          VBUSDATA        <^X1E>,1                ;1C
             01 0271 11152          .BYTE   ^X01
                0272 11153          VBUSDATA        <^X1E>,1                ;1D
```

```
01   0273 11154          .BYTE   ^X01
     0274 11155          VBUSDATA        <^X1E>,1                ;1E
01   0275 11156          .BYTE   ^X01
     0276 11157          VBUSDATA        <^X1E>,1                ;1F
01   0277 11158          .BYTE   ^X01
     0278 11159          VBUSDATA        <^X1E>,1                ;20
01   0279 11160          .BYTE   ^X01
     027A 11161          VBUSDATA        <^X1E>,1                ;21
01   027B 11162          .BYTE   ^X01
     027C 11163          VBUSDATA        <^X1E>,1                ;22
01   027D 11164          .BYTE   ^X01
     027E 11165          VBUSDATA        <^X1E>,1                ;23
01   027F 11166          .BYTE   ^X01
     0280 11167          VBUSDATA        <^X1E>,1                ;24
01   0281 11168          .BYTE   ^X01
     0282 11169          VBUSDATA        <^X1E>,1                ;25
01   0283 11170          .BYTE   ^X01
     0284 11171          VBUSDATA        <^X1E>,1                ;26
01   0285 11172          .BYTE   ^X01
     0286 11173          VBUSDATA        <^X1E>,1                ;27
01   0287 11174          .BYTE   ^X01
     0288 11175          VBUSDATA        <^X1E>,1                ;28
01   0289 11176          .BYTE   ^X01
     028A 11177          VBUSDATA        <^X1E>,1                ;29
01   028B 11178          .BYTE   ^X01
     028C 11179          VBUSDATA        <^X1E>,1                ;2A
01   028D 11180          .BYTE   ^X01
     028E 11181          VBUSDATA        <^X1E>,1                ;2B
01   028F 11182          .BYTE   ^X01
     0290 11183          VBUSDATA        <^X1E>,1                ;2C
01   0291 11184          .BYTE   ^X01
     0292 11185          VBUSDATA        <^X1E>,1                ;2D
01   0293 11186          .BYTE   ^X01
     0294 11187          VBUSDATA        <^X1E>,1                ;2E
01   0295 11188          .BYTE   ^X01
     0296 11189          VBUSDATA        <^X1E>,1                ;2F
01   0297 11190          .BYTE   ^X01
     0298 11191          VBUSDATA        <^X1E>,1                ;30
01   0299 11192          .BYTE   ^X01
     029A 11193          VBUSDATA        <^X1E>,1                ;31
01   029B 11194          .BYTE   ^X01
     029C 11195          VBUSDATA        <^X1E>,1                ;32
01   029D 11196          .BYTE   ^X01
     029E 11197          VBUSDATA        <^X1E>,1                ;33
01   029F 11198          .BYTE   ^X01
     02A0 11199          VBUSDATA        <^X1E>,1                ;34
01   02A1 11200          .BYTE   ^X01
     02A2 11201          VBUSDATA        <^X1E>,1                ;35
01   02A3 11202          .BYTE   ^X01
     02A4 11203          VBUSDATA        <^X1E>,1                ;36
01   02A5 11204          .BYTE   ^X01
     02A6 11205          VBUSDATA        <^X1E>,1                ;37
01   02A7 11206          .BYTE   ^X01
     02A8 11207          VBUSDATA        <^X1E>,1                ;38
01   02A9 11208          .BYTE   ^X01
```

```
                02AA 11209          VBUSDATA        <^X1E>,1                ;39
          01    02AB 11210          .BYTE   ^X01
                02AC 11211          VBUSDATA        <^X1E>,1                ;3A
          01    02AD 11212          .BYTE   ^X01
                02AE 11213          VBUSDATA        <^X1E>,1                ;3B
          01    02AF 11214          .BYTE   ^X01
                02B0 11215          VBUSDATA        <^X1E>,1                ;3C
          01    02B1 11216          .BYTE   ^X01
                02B2 11217          VBUSDATA        <^X1E>,0                ;3D
          01    02B3 11218          .BYTE   ^X01
                02B4 11219          VBUSDATA        <^X1E>,1                ;3E
          01    02B5 11220          .BYTE   ^X01
                02B6 11221          VBUSDATA        <^X1E>,1                ;3F
          01    02B7 11222          .BYTE   ^X01
                02B8 11223          VBUSDATA        <^X1E>,1                ;40
          01    02B9 11224          .BYTE   ^X01
                02BA 11225          VBUSDATA        <^X1E>,1                ;41
          01    02BB 11226          .BYTE   ^X01
                02BC 11227          VBUSDATA        <^X1E>,1                ;42
          01    02BD 11228          .BYTE   ^X01
                02BE 11229          VBUSDATA        <^X1E>,1                ;43
          01    02BF 11230          .BYTE   ^X01
                02C0 11231          VBUSDATA        <^X1E>,1                ;44
          01    02C1 11232          .BYTE   ^X01
                02C2 11233          VBUSDATA        <^X1E>,1                ;45
          01    02C3 11234          .BYTE   ^X01
                02C4 11235          VBUSDATA        <^X1E>,1                ;46
          01    02C5 11236          .BYTE   ^X01
                02C6 11237          VBUSDATA        <^X1E>,1                ;47
          01    02C7 11238          .BYTE   ^X01
                02C8 11239          VBUSDATA        <^X1E>,1                ;48
          01    02C9 11240          .BYTE   ^X01
                02CA 11241          VBUSDATA        <^X1E>,1                ;49
          01    02CB 11242          .BYTE   ^X01
                02CC 11243          VBUSDATA        <^X1E>,1                ;4A
          01    02CD 11244          .BYTE   ^X01
                02CE 11245          VBUSDATA        <^X1E>,1                ;4B
          01    02CF 11246          .BYTE   ^X01
                02D0 11247          VBUSDATA        <^X1E>,1                ;4C
          01    02D1 11248          .BYTE   ^X01
                02D2 11249          VBUSDATA        <^X1E>,1                ;4D
          01    02D3 11250          .BYTE   ^X01
                02D4 11251          VBUSDATA        <^X1E>,1                ;4E
          01    02D5 11252          .BYTE   ^X01
                02D6 11253          VBUSDATA        <^X1E>,1                ;4F
          01    02D7 11254          .BYTE   ^X01
                02D8 11255          VBUSDATA        <^X1E>,1                ;50
          01    02D9 11256          .BYTE   ^X01
                02DA 11257          VBUSDATA        <^X1E>,1                ;51
          01    02DB 11258          .BYTE   ^X01
                02DC 11259          VBUSDATA        <^X1E>,1                ;52
          01    02DD 11260          .BYTE   ^X01
                02DE 11261          VBUSDATA        <^X1E>,1                ;53
          01    02DF 11262          .BYTE   ^X01
                02E0 11263          VBUSDATA        <^X1E>,1                ;54
```

```
                01  02E1 11264          .BYTE   ^X01
                    02E2 11265          VBUSDATA        <^X1E>,1                ;55
                01  02E3 11266          .BYTE   ^X01
                    02E4 11267          VBUSDATA        <^X1E>,1                ;56
                01  02E5 11268          .BYTE   ^X01
                    02E6 11269          VBUSDATA        <^X1E>,1                ;57
                01  02E7 11270          .BYTE   ^X01
                    02E8 11271          VBUSDATA        <^X1E>,1                ;58
                01  02E9 11272          .BYTE   ^X01
                    02EA 11273          VBUSDATA        <^X1E>,1                ;59
                01  02EB 11274          .BYTE   ^X01
                    02EC 11275          VBUSDATA        <^X1E>,1                ;5A
                01  02ED 11276          .BYTE   ^X01
                    02EE 11277          VBUSDATA        <^X1E>,1                ;5B
                01  02EF 11278          .BYTE   ^X01
                    02F0 11279          VBUSDATA        <^X1E>,1                ;5C
                01  02F1 11280          .BYTE   ^X01
                    02F2 11281          VBUSDATA        <^X1E>,1                ;5D
                01  02F3 11282          .BYTE   ^X01
                    02F4 11283          VBUSDATA        <^X1E>,1                ;5E
                01  02F5 11284          .BYTE   ^X01
                    02F6 11285          VBUSDATA        <^X1E>,1                ;5F
                01  02F7 11286          .BYTE   ^X01
                    02F8 11287          VBUSDATA        <^X1E>,1                ;60
                01  02F9 11288          .BYTE   ^X01
                    02FA 11289          VBUSDATA        <^X1E>,1                ;61
                01  02FB 11290          .BYTE   ^X01
                    02FC 11291          VBUSDATA        <^X1E>,1                ;62
                01  02FD 11292          .BYTE   ^X01
                    02FE 11293          VBUSDATA        <^X1E>,1                ;63
                01  02FF 11294          .BYTE   ^X01
                    0300 11295          VBUSDATA        <^X1E>,1                ;64
                01  0301 11296          .BYTE   ^X01
                    0302 11297          VBUSDATA        <^X1E>,1                ;65
                01  0303 11298          .BYTE   ^X01
                    0304 11299          VBUSDATA        <^X1E>,1                ;66
                01  0305 11300          .BYTE   ^X01
                    0306 11301          VBUSDATA        <^X1E>,1                ;67
                01  0307 11302          .BYTE   ^X01
                    0308 11303          VBUSDATA        <^X1E>,1                ;68
                01  0309 11304          .BYTE   ^X01
                    030A 11305          VBUSDATA        <^X1E>,1                ;69
                01  030B 11306          .BYTE   ^X01
                    030C 11307          VBUSDATA        <^X1E>,1                ;6A
                01  030D 11308          .BYTE   ^X01
                    030E 11309          VBUSDATA        <^X1E>,1                ;6B
                01  030F 11310          .BYTE   ^X01
                    0310 11311          VBUSDATA        <^X1E>,1                ;6C
                01  0311 11312          .BYTE   ^X01
                    0312 11313          VBUSDATA        <^X1E>,1                ;6D
                01  0313 11314          .BYTE   ^X01
                    0314 11315          VBUSDATA        <^X1E>,1                ;6E
                01  0315 11316          .BYTE   ^X01
                    0316 11317          VBUSDATA        <^X1E>,1                ;6F
                01  0317 11318          .BYTE   ^X01
```

```
                 0318 11319          VBUSDATA        <^X1E>,1                ;70
01               0319 11320          .BYTE   ^X01
                 031A 11321          VBUSDATA        <^X1E>,1                ;71
01               031B 11322          .BYTE   ^X01
                 031C 11323          VBUSDATA        <^X1E>,1                ;72
01               031D 11324          .BYTE   ^X01
                 031E 11325          VBUSDATA        <^X1E>,1                ;73
01               031F 11326          .BYTE   ^X01
                 0320 11327          VBUSDATA        <^X1E>,1                ;74
01               0321 11328          .BYTE   ^X01
                 0322 11329          VBUSDATA        <^X1E>,1                ;75
01               0323 11330          .BYTE   ^X01
                 0324 11331          VBUSDATA        <^X1E>,1                ;76
01               0325 11332          .BYTE   ^X01
                 0326 11333          VBUSDATA        <^X1E>,1                ;77
01               0327 11334          .BYTE   ^X01
                 0328 11335          VBUSDATA        <^X1E>,1                ;78
01               0329 11336          .BYTE   ^X01
                 032A 11337          VBUSDATA        <^X1E>,1                ;79
01               032B 11338          .BYTE   ^X01
                 032C 11339          VBUSDATA        <^X1E>,1                ;7A
01               032D 11340          .BYTE   ^X01
                 032E 11341          VBUSDATA        <^X1E>,1                ;7B
01               032F 11342          .BYTE   ^X01
                 0330 11343          VBUSDATA        <^X1E>,1                ;7C
01               0331 11344          .BYTE   ^X01
                 0332 11345          VBUSDATA        <^X1E>,1                ;7D
01               0333 11346          .BYTE   ^X01
                 0334 11347          VBUSDATA        <^X1E>,1                ;7E
01               0335 11348          .BYTE   ^X01
                 0336 11349          VBUSDATA        <^X1E>,1                ;7F
01               0337 11350          .BYTE   ^X01
                 0338 11351          VBUSDATA        <^X1E>,1                ;80
01               0339 11352          .BYTE   ^X01
                 033A 11353          VBUSDATA        <^X1E>,0                ;81
01               033B 11354          .BYTE   ^X01
                 033C 11355          VBUSDATA        <^X1E>,1                ;82
01               033D 11356          .BYTE   ^X01
                 033E 11357          VBUSDATA        <^X1E>,0                ;83
01               033F 11358          .BYTE   ^X01
                 0340 11359          VBUSDATA        <^X1E>,1                ;84
01               0341 11360          .BYTE   ^X01
                 0342 11361          VBUSDATA        <^X1E>,1                ;85
01               0343 11362          .BYTE   ^X01
                 0344 11363          VBUSDATA        <^X1E>,1                ;86
01               0345 11364          .BYTE   ^X01
                 0346 11365          VBUSDATA        <^X1E>,1                ;87
01               0347 11366          .BYTE   ^X01
                 0348 11367          VBUSDATA        <^X1E>,1                ;88
01               0349 11368          .BYTE   ^X01
                 034A 11369          VBUSDATA        <^X1E>,0                ;89
01               034B 11370          .BYTE   ^X01
                 034C 11371          VBUSDATA        <^X1E>,1                ;8A
01               034D 11372          .BYTE   ^X01
                 034E 11373          VBUSDATA        <^X1E>,0                ;8B
```

```
01  034F 11374          .BYTE   ^X01
    0350 11375          VBUSDATA        <^X1E>,1                ;8C
01  0351 11376          .BYTE   ^X01
    0352 11377          VBUSDATA        <^X1E>,0                ;8D
01  0353 11378          .BYTE   ^X01
    0354 11379          VBUSDATA        <^X1E>,1                ;8E
01  0355 11380          .BYTE   ^X01
    0356 11381          VBUSDATA        <^X1E>,0                ;8F
01  0357 11382          .BYTE   ^X01
    0358 11383          VBUSDATA        <^X1E>,1                ;90
01  0359 11384          .BYTE   ^X01
    035A 11385          VBUSDATA        <^X1E>,1                ;91
01  035B 11386          .BYTE   ^X01
    035C 11387          VBUSDATA        <^X1E>,1                ;92
01  035D 11388          .BYTE   ^X01
    035E 11389          VBUSDATA        <^X1E>,1                ;93
01  035F 11390          .BYTE   ^X01
    0360 11391          VBUSDATA        <^X1E>,1                ;94
01  0361 11392          .BYTE   ^X01
    0362 11393          VBUSDATA        <^X1E>,1                ;95
01  0363 11394          .BYTE   ^X01
    0364 11395          VBUSDATA        <^X1E>,1                ;96
01  0365 11396          .BYTE   ^X01
    0366 11397          VBUSDATA        <^X1E>,1                ;97
01  0367 11398          .BYTE   ^X01
    0368 11399          VBUSDATA        <^X1E>,1                ;98
01  0369 11400          .BYTE   ^X01
    036A 11401          VBUSDATA        <^X1E>,1                ;99
01  036B 11402          .BYTE   ^X01
    036C 11403          VBUSDATA        <^X1E>,1                ;9A
01  036D 11404          .BYTE   ^X01
    036E 11405          VBUSDATA        <^X1E>,1                ;9B
01  036F 11406          .BYTE   ^X01
    0370 11407          VBUSDATA        <^X1E>,0                ;9C
01  0371 11408          .BYTE   ^X01
    0372 11409          VBUSDATA        <^X1E>,0                ;9D
01  0373 11410          .BYTE   ^X01
    0374 11411          VBUSDATA        <^X1E>,1                ;9E
01  0375 11412          .BYTE   ^X01
    0376 11413          VBUSDATA        <^X1E>,1                ;9F
01  0377 11414          .BYTE   ^X01
    0378 11415          VBUSDATA        <^X1E>,1                ;A0
01  0379 11416          .BYTE   ^X01
    037A 11417          VBUSDATA        <^X1E>,0                ;A1
01  037B 11418          .BYTE   ^X01
    037C 11419          VBUSDATA        <^X1E>,1                ;A2
01  037D 11420          .BYTE   ^X01
    037E 11421          VBUSDATA        <^X1E>,0                ;A3
01  037F 11422          .BYTE   ^X01
    0380 11423          VBUSDATA        <^X1E>,1                ;A4
01  0381 11424          .BYTE   ^X01
    0382 11425          VBUSDATA        <^X1E>,1                ;A5
01  0383 11426          .BYTE   ^X01
    0384 11427          VBUSDATA        <^X1E>,1                ;A6
01  0385 11428          .BYTE   ^X01
```

```
                    0386 11429          VBUSDATA        <^X1E>,1                ;A7
01                  0387 11430          .BYTE   ^X01
                    0388 11431          VBUSDATA        <^X1E>,1                ;A8
01                  0389 11432          .BYTE   ^X01
                    038A 11433          VBUSDATA        <^X1E>,0                ;A9
01                  038B 11434          .BYTE   ^X01
                    038C 11435          VBUSDATA        <^X1E>,1                ;AA
01                  038D 11436          .BYTE   ^X01
                    038E 11437          VBUSDATA        <^X1E>,0                ;AB
01                  038F 11438          .BYTE   ^X01
                    0390 11439          VBUSDATA        <^X1E>,1                ;AC
01                  0391 11440          .BYTE   ^X01
                    0392 11441          VBUSDATA        <^X1E>,0                ;AD
01                  0393 11442          .BYTE   ^X01
                    0394 11443          VBUSDATA        <^X1E>,1                ;AE
01                  0395 11444          .BYTE   ^X01
                    0396 11445          VBUSDATA        <^X1E>,0                ;AF
01                  0397 11446          .BYTE   ^X01
                    0398 11447          VBUSDATA        <^X1E>,1                ;B0
01                  0399 11448          .BYTE   ^X01
                    039A 11449          VBUSDATA        <^X1E>,1                ;B1
01                  039B 11450          .BYTE   ^X01
                    039C 11451          VBUSDATA        <^X1E>,1                ;B2
01                  039D 11452          .BYTE   ^X01
                    039E 11453          VBUSDATA        <^X1E>,1                ;B3
01                  039F 11454          .BYTE   ^X01
                    03A0 11455          VBUSDATA        <^X1E>,1                ;B4
01                  03A1 11456          .BYTE   ^X01
                    03A2 11457          VBUSDATA        <^X1E>,1                ;B5
01                  03A3 11458          .BYTE   ^X01
                    03A4 11459          VBUSDATA        <^X1E>,1                ;B6
01                  03A5 11460          .BYTE   ^X01
                    03A6 11461          VBUSDATA        <^X1E>,1                ;B7
01                  03A7 11462          .BYTE   ^X01
                    03A8 11463          VBUSDATA        <^X1E>,1                ;B8
01                  03A9 11464          .BYTE   ^X01
                    03AA 11465          VBUSDATA        <^X1E>,1                ;B9
01                  03AB 11466          .BYTE   ^X01
                    03AC 11467          VBUSDATA        <^X1E>,1                ;BA
01                  03AD 11468          .BYTE   ^X01
                    03AE 11469          VBUSDATA        <^X1E>,1                ;BB
01                  03AF 11470          .BYTE   ^X01
                    03B0 11471          VBUSDATA        <^X1E>,1                ;BC
01                  03B1 11472          .BYTE   ^X01
                    03B2 11473          VBUSDATA        <^X1E>,1                ;BD
01                  03B3 11474          .BYTE   ^X01
                    03B4 11475          VBUSDATA        <^X1E>,1                ;BE
01                  03B5 11476          .BYTE   ^X01
                    03B6 11477          VBUSDATA        <^X1E>,1                ;BF
01                  03B7 11478          .BYTE   ^X01
                    03B8 11479          VBUSDATA        <^X1E>,1                ;C0
01                  03B9 11480          .BYTE   ^X01
                    03BA 11481          VBUSDATA        <^X1E>,0                ;C1
01                  03BB 11482          .BYTE   ^X01
                    03BC 11483          VBUSDATA        <^X1E>,1                ;C2
```

```
                    01  03BD 11484          .BYTE   ^X01
                        03BE 11485          VBUSDATA        <^X1E>,0                ;C3
                    01  03BF 11486          .BYTE   ^X01
                        03C0 11487          VBUSDATA        <^X1E>,1                ;C4
                    01  03C1 11488          .BYTE   ^X01
                        03C2 11489          VBUSDATA        <^X1E>,1                ;C5
                    01  03C3 11490          .BYTE   ^X01
                        03C4 11491          VBUSDATA        <^X1E>,1                ;C6
                    01  03C5 11492          .BYTE   ^X01
                        03C6 11493          VBUSDATA        <^X1E>,1                ;C7
                    01  03C7 11494          .BYTE   ^X01
                        03C8 11495          VBUSDATA        <^X1E>,1                ;C8
                    01  03C9 11496          .BYTE   ^X01
                        03CA 11497          VBUSDATA        <^X1E>,0                ;C9
                    01  03CB 11498          .BYTE   ^X01
                        03CC 11499          VBUSDATA        <^X1E>,1                ;CA
                    01  03CD 11500          .BYTE   ^X01
                        03CE 11501          VBUSDATA        <^X1E>,0                ;CB
                    01  03CF 11502          .BYTE   ^X01
                        03D0 11503          VBUSDATA        <^X1E>,1                ;CC
                    01  03D1 11504          .BYTE   ^X01
                        03D2 11505          VBUSDATA        <^X1E>,0                ;CD
                    01  03D3 11506          .BYTE   ^X01
                        03D4 11507          VBUSDATA        <^X1E>,1                ;CE
                    01  03D5 11508          .BYTE   ^X01
                        03D6 11509          VBUSDATA        <^X1E>,0                ;CF
                    01  03D7 11510          .BYTE   ^X01
                        03D8 11511          VBUSDATA        <^X1E>,1                ;D0
                    01  03D9 11512          .BYTE   ^X01
                        03DA 11513          VBUSDATA        <^X1E>,1                ;D1
                    01  03DB 11514          .BYTE   ^X01
                        03DC 11515          VBUSDATA        <^X1E>,1                ;D2
                    01  03DD 11516          .BYTE   ^X01
                        03DE 11517          VBUSDATA        <^X1E>,1                ;D3
                    01  03DF 11518          .BYTE   ^X01
                        03E0 11519          VBUSDATA        <^X1E>,1                ;D4
                    01  03E1 11520          .BYTE   ^X01
                        03E2 11521          VBUSDATA        <^X1E>,1                ;D5
                    01  03E3 11522          .BYTE   ^X01
                        03E4 11523          VBUSDATA        <^X1E>,1                ;D6
                    01  03E5 11524          .BYTE   ^X01
                        03E6 11525          VBUSDATA        <^X1E>,1                ;D7
                    01  03E7 11526          .BYTE   ^X01
                        03E8 11527          VBUSDATA        <^X1E>,1                ;D8
                    01  03E9 11528          .BYTE   ^X01
                        03EA 11529          VBUSDATA        <^X1E>,1                ;D9
                    01  03EB 11530          .BYTE   ^X01
                        03EC 11531          VBUSDATA        <^X1E>,1                ;DA
                    01  03ED 11532          .BYTE   ^X01
                        03EE 11533          VBUSDATA        <^X1E>,1                ;DB
                    01  03EF 11534          .BYTE   ^X01
                        03F0 11535          VBUSDATA        <^X1E>,1                ;DC
                    01  03F1 11536          .BYTE   ^X01
                        03F2 11537          VBUSDATA        <^X1E>,1                ;DD
                    01  03F3 11538          .BYTE   ^X01
```

```
                    03F4 11539          VBUSDATA        <^X1E>,1                ;DE
                01  03F5 11540          .BYTE   ^X01
                    03F6 11541          VBUSDATA        <^X1E>,1                ;DF
                01  03F7 11542          .BYTE   ^X01
                    03F8 11543          VBUSDATA        <^X1E>,1                ;E0
                01  03F9 11544          .BYTE   ^X01
                    03FA 11545          VBUSDATA        <^X1E>,1                ;E1
                01  03FB 11546          .BYTE   ^X01
                    03FC 11547          VBUSDATA        <^X1E>,1                ;E2
                01  03FD 11548          .BYTE   ^X01
                    03FE 11549          VBUSDATA        <^X1E>,1                ;E3
                01  03FF 11550          .BYTE   ^X01
                    0400 11551          VBUSDATA        <^X1E>,1                ;E4
                01  0401 11552          .BYTE   ^X01
                    0402 11553          VBUSDATA        <^X1E>,1                ;E5
                01  0403 11554          .BYTE   ^X01
                    0404 11555          VBUSDATA        <^X1E>,1                ;E6
                01  0405 11556          .BYTE   ^X01
                    0406 11557          VBUSDATA        <^X1E>,1                ;E7
                01  0407 11558          .BYTE   ^X01
                    0408 11559          VBUSDATA        <^X1E>,1                ;E8
                01  0409 11560          .BYTE   ^X01
                    040A 11561          VBUSDATA        <^X1E>,1                ;E9
                01  040B 11562          .BYTE   ^X01
                    040C 11563          VBUSDATA        <^X1E>,0                ;EA
                01  040D 11564          .BYTE   ^X01
                    040E 11565          VBUSDATA        <^X1E>,0                ;EB
                01  040F 11566          .BYTE   ^X01
                    0410 11567          VBUSDATA        <^X1E>,0                ;EC
                01  0411 11568          .BYTE   ^X01
                    0412 11569          VBUSDATA        <^X1E>,0                ;ED
                01  0413 11570          .BYTE   ^X01
                    0414 11571          VBUSDATA        <^X1E>,0                ;EE
                01  0415 11572          .BYTE   ^X01
                    0416 11573          VBUSDATA        <^X1E>,0                ;EF
                01  0417 11574          .BYTE   ^X01
                    0418 11575          VBUSDATA        <^X1E>,1                ;F0
                01  0419 11576          .BYTE   ^X01
                    041A 11577          VBUSDATA        <^X1E>,0                ;F1
                01  041B 11578          .BYTE   ^X01
                    041C 11579          VBUSDATA        <^X1E>,1                ;F2
                01  041D 11580          .BYTE   ^X01
                    041E 11581          VBUSDATA        <^X1E>,1                ;F3
                01  041F 11582          .BYTE   ^X01
                    0420 11583          VBUSDATA        <^X1E>,1                ;F4
                01  0421 11584          .BYTE   ^X01
                    0422 11585          VBUSDATA        <^X1E>,1                ;F5
                01  0423 11586          .BYTE   ^X01
                    0424 11587          VBUSDATA        <^X1E>,1                ;F6
                01  0425 11588          .BYTE   ^X01
                    0426 11589          VBUSDATA        <^X1E>,1                ;F7
                01  0427 11590          .BYTE   ^X01
                    0428 11591          VBUSDATA        <^X1E>,1                ;F8
                01  0429 11592          .BYTE   ^X01
                    042A 11593          VBUSDATA        <^X1E>,1                ;F9
```

```
                01  042B 11594          .BYTE   ^X01
                    042C 11595          VBUSDATA        <^X1E>,1                ;FA
                01  042D 11596          .BYTE   ^X01
                    042E 11597          VBUSDATA        <^X1E>,1                ;FB
                01  042F 11598          .BYTE   ^X01
                    0430 11599          VBUSDATA        <^X1E>,1                ;FC
                01  0431 11600          .BYTE   ^X01
                    0432 11601          VBUSDATA        <^X1E>,1                ;FD
                01  0433 11602          .BYTE   ^X01
                    0434 11603          VBUSDATA        <^X1E>,1                ;FE
                01  0435 11604          .BYTE   ^X01
                    0436 11605          VBUSDATA        <^X1E>,1                ;FF
                    0437 11606
                    0437 11607 ;
                    0437 11608 ; Expected VBUS values for Subtest 3
                    0437 11609 ;
                    0437 11610
                01  0437 11611 31$:     .BYTE   ^X01
                    0438 11612          VBUSDATA        <^X1E>,1                ;00
                01  0439 11613          .BYTE   ^X01
                    043A 11614          VBUSDATA        <^X1E>,1                ;01
                01  043B 11615          .BYTE   ^X01
                    043C 11616          VBUSDATA        <^X1E>,1                ;02
                01  043D 11617          .BYTE   ^X01
                    043E 11618          VBUSDATA        <^X1E>,1                ;03
                01  043F 11619          .BYTE   ^X01
                    0440 11620          VBUSDATA        <^X1E>,1                ;04
                01  0441 11621          .BYTE   ^X01
                    0442 11622          VBUSDATA        <^X1E>,1                ;05
                01  0443 11623          .BYTE   ^X01
                    0444 11624          VBUSDATA        <^X1E>,1                ;06
                01  0445 11625          .BYTE   ^X01
                    0446 11626          VBUSDATA        <^X1E>,1                ;07
                01  0447 11627          .BYTE   ^X01
                    0448 11628          VBUSDATA        <^X1E>,1                ;08
                01  0449 11629          .BYTE   ^X01
                    044A 11630          VBUSDATA        <^X1E>,1                ;09
                01  044B 11631          .BYTE   ^X01
                    044C 11632          VBUSDATA        <^X1E>,1                ;0A
                01  044D 11633          .BYTE   ^X01
                    044E 11634          VBUSDATA        <^X1E>,1                ;0B
                01  044F 11635          .BYTE   ^X01
                    0450 11636          VBUSDATA        <^X1E>,1                ;0C
                01  0451 11637          .BYTE   ^X01
                    0452 11638          VBUSDATA        <^X1E>,1                ;0D
                01  0453 11639          .BYTE   ^X01
                    0454 11640          VBUSDATA        <^X1E>,1                ;0E
                01  0455 11641          .BYTE   ^X01
                    0456 11642          VBUSDATA        <^X1E>,1                ;0F
                01  0457 11643          .BYTE   ^X01
                    0458 11644          VBUSDATA        <^X1E>,1                ;10
                01  0459 11645          .BYTE   ^X01
                    045A 11646          VBUSDATA        <^X1E>,1                ;11
                01  045B 11647          .BYTE   ^X01
                    045C 11648          VBUSDATA        <^X1E>,1                ;12
```

```
01  045D 11649          .BYTE   ^X01
    045E 11650          VBUSDATA        <^X1E>,1                ;13
01  045F 11651          .BYTE   ^X01
    0460 11652          VBUSDATA        <^X1E>,1                ;14
01  0461 11653          .BYTE   ^X01
    0462 11654          VBUSDATA        <^X1E>,1                ;15
01  0463 11655          .BYTE   ^X01
    0464 11656          VBUSDATA        <^X1E>,1                ;16
01  0465 11657          .BYTE   ^X01
    0466 11658          VBUSDATA        <^X1E>,1                ;17
01  0467 11659          .BYTE   ^X01
    0468 11660          VBUSDATA        <^X1E>,1                ;18
01  0469 11661          .BYTE   ^X01
    046A 11662          VBUSDATA        <^X1E>,1                ;19
01  046B 11663          .BYTE   ^X01
    046C 11664          VBUSDATA        <^X1E>,1                ;1A
01  046D 11665          .BYTE   ^X01
    046E 11666          VBUSDATA        <^X1E>,1                ;1B
01  046F 11667          .BYTE   ^X01
    0470 11668          VBUSDATA        <^X1E>,1                ;1C
01  0471 11669          .BYTE   ^X01
    0472 11670          VBUSDATA        <^X1E>,1                ;1D
01  0473 11671          .BYTE   ^X01
    0474 11672          VBUSDATA        <^X1E>,1                ;1E
01  0475 11673          .BYTE   ^X01
    0476 11674          VBUSDATA        <^X1E>,1                ;1F
01  0477 11675          .BYTE   ^X01
    0478 11676          VBUSDATA        <^X1E>,1                ;20
01  0479 11677          .BYTE   ^X01
    047A 11678          VBUSDATA        <^X1E>,1                ;21
01  047B 11679          .BYTE   ^X01
    047C 11680          VBUSDATA        <^X1E>,1                ;22
01  047D 11681          .BYTE   ^X01
    047E 11682          VBUSDATA        <^X1E>,1                ;23
01  047F 11683          .BYTE   ^X01
    0480 11684          VBUSDATA        <^X1E>,1                ;24
01  0481 11685          .BYTE   ^X01
    0482 11686          VBUSDATA        <^X1E>,1                ;25
01  0483 11687          .BYTE   ^X01
    0484 11688          VBUSDATA        <^X1E>,1                ;26
01  0485 11689          .BYTE   ^X01
    0486 11690          VBUSDATA        <^X1E>,1                ;27
01  0487 11691          .BYTE   ^X01
    0488 11692          VBUSDATA        <^X1E>,1                ;28
01  0489 11693          .BYTE   ^X01
    048A 11694          VBUSDATA        <^X1E>,1                ;29
01  048B 11695          .BYTE   ^X01
    048C 11696          VBUSDATA        <^X1E>,1                ;2A
01  048D 11697          .BYTE   ^X01
    048E 11698          VBUSDATA        <^X1E>,1                ;2B
01  048F 11699          .BYTE   ^X01
    0490 11700          VBUSDATA        <^X1E>,1                ;2C
01  0491 11701          .BYTE   ^X01
    0492 11702          VBUSDATA        <^X1E>,1                ;2D
01  0493 11703          .BYTE   ^X01
```

```
                    0494 11704          VBUSDATA        <^X1E>,1                ;2E
01                  0495 11705          .BYTE   ^X01
                    0496 11706          VBUSDATA        <^X1E>,1                ;2F
01                  0497 11707          .BYTE   ^X01
                    0498 11708          VBUSDATA        <^X1E>,1                ;30
01                  0499 11709          .BYTE   ^X01
                    049A 11710          VBUSDATA        <^X1E>,1                ;31
01                  049B 11711          .BYTE   ^X01
                    049C 11712          VBUSDATA        <^X1E>,1                ;32
01                  049D 11713          .BYTE   ^X01
                    049E 11714          VBUSDATA        <^X1E>,1                ;33
01                  049F 11715          .BYTE   ^X01
                    04A0 11716          VBUSDATA        <^X1E>,1                ;34
01                  04A1 11717          .BYTE   ^X01
                    04A2 11718          VBUSDATA        <^X1E>,1                ;35
01                  04A3 11719          .BYTE   ^X01
                    04A4 11720          VBUSDATA        <^X1E>,1                ;36
01                  04A5 11721          .BYTE   ^X01
                    04A6 11722          VBUSDATA        <^X1E>,1                ;37
01                  04A7 11723          .BYTE   ^X01
                    04A8 11724          VBUSDATA        <^X1E>,1                ;38
01                  04A9 11725          .BYTE   ^X01
                    04AA 11726          VBUSDATA        <^X1E>,1                ;39
01                  04AB 11727          .BYTE   ^X01
                    04AC 11728          VBUSDATA        <^X1E>,1                ;3A
01                  04AD 11729          .BYTE   ^X01
                    04AE 11730          VBUSDATA        <^X1E>,1                ;3B
01                  04AF 11731          .BYTE   ^X01
                    04B0 11732          VBUSDATA        <^X1E>,1                ;3C
01                  04B1 11733          .BYTE   ^X01
                    04B2 11734          VBUSDATA        <^X1E>,0                ;3D
01                  04B3 11735          .BYTE   ^X01
                    04B4 11736          VBUSDATA        <^X1E>,1                ;3E
01                  04B5 11737          .BYTE   ^X01
                    04B6 11738          VBUSDATA        <^X1E>,1                ;3F
01                  04B7 11739          .BYTE   ^X01
                    04B8 11740          VBUSDATA        <^X1E>,1                ;40
01                  04B9 11741          .BYTE   ^X01
                    04BA 11742          VBUSDATA        <^X1E>,1                ;41
01                  04BB 11743          .BYTE   ^X01
                    04BC 11744          VBUSDATA        <^X1E>,1                ;42
01                  04BD 11745          .BYTE   ^X01
                    04BE 11746          VBUSDATA        <^X1E>,1                ;43
01                  04BF 11747          .BYTE   ^X01
                    04C0 11748          VBUSDATA        <^X1E>,1                ;44
01                  04C1 11749          .BYTE   ^X01
                    04C2 11750          VBUSDATA        <^X1E>,1                ;45
01                  04C3 11751          .BYTE   ^X01
                    04C4 11752          VBUSDATA        <^X1E>,1                ;46
01                  04C5 11753          .BYTE   ^X01
                    04C6 11754          VBUSDATA        <^X1E>,1                ;47
01                  04C7 11755          .BYTE   ^X01
                    04C8 11756          VBUSDATA        <^X1E>,1                ;48
01                  04C9 11757          .BYTE   ^X01
                    04CA 11758          VBUSDATA        <^X1E>,1                ;49
```

```
                01  04CB 11759          .BYTE   ^X01
                    04CC 11760          VBUSDATA        <^X1E>,1                ;4A
                01  04CD 11761          .BYTE   ^X01
                    04CE 11762          VBUSDATA        <^X1E>,1                ;4B
                01  04CF 11763          .BYTE   ^X01
                    04D0 11764          VBUSDATA        <^X1E>,1                ;4C
                01  04D1 11765          .BYTE   ^X01
                    04D2 11766          VBUSDATA        <^X1E>,1                ;4D
                01  04D3 11767          .BYTE   ^X01
                    04D4 11768          VBUSDATA        <^X1E>,1                ;4E
                01  04D5 11769          .BYTE   ^X01
                    04D6 11770          VBUSDATA        <^X1E>,1                ;4F
                01  04D7 11771          .BYTE   ^X01
                    04D8 11772          VBUSDATA        <^X1E>,1                ;50
                01  04D9 11773          .BYTE   ^X01
                    04DA 11774          VBUSDATA        <^X1E>,1                ;51
                01  04DB 11775          .BYTE   ^X01
                    04DC 11776          VBUSDATA        <^X1E>,1                ;52
                01  04DD 11777          .BYTE   ^X01
                    04DE 11778          VBUSDATA        <^X1E>,1                ;53
                01  04DF 11779          .BYTE   ^X01
                    04E0 11780          VBUSDATA        <^X1E>,1                ;54
                01  04E1 11781          .BYTE   ^X01
                    04E2 11782          VBUSDATA        <^X1E>,1                ;55
                01  04E3 11783          .BYTE   ^X01
                    04E4 11784          VBUSDATA        <^X1E>,1                ;56
                01  04E5 11785          .BYTE   ^X01
                    04E6 11786          VBUSDATA        <^X1E>,1                ;57
                01  04E7 11787          .BYTE   ^X01
                    04E8 11788          VBUSDATA        <^X1E>,1                ;58
                01  04E9 11789          .BYTE   ^X01
                    04EA 11790          VBUSDATA        <^X1E>,1                ;59
                01  04EB 11791          .BYTE   ^X01
                    04EC 11792          VBUSDATA        <^X1E>,1                ;5A
                01  04ED 11793          .BYTE   ^X01
                    04EE 11794          VBUSDATA        <^X1E>,1                ;5B
                01  04EF 11795          .BYTE   ^X01
                    04F0 11796          VBUSDATA        <^X1E>,1                ;5C
                01  04F1 11797          .BYTE   ^X01
                    04F2 11798          VBUSDATA        <^X1E>,1                ;5D
                01  04F3 11799          .BYTE   ^X01
                    04F4 11800          VBUSDATA        <^X1E>,1                ;5E
                01  04F5 11801          .BYTE   ^X01
                    04F6 11802          VBUSDATA        <^X1E>,1                ;5F
                01  04F7 11803          .BYTE   ^X01
                    04F8 11804          VBUSDATA        <^X1E>,1                ;60
                01  04F9 11805          .BYTE   ^X01
                    04FA 11806          VBUSDATA        <^X1E>,1                ;61
                01  04FB 11807          .BYTE   ^X01
                    04FC 11808          VBUSDATA        <^X1E>,1                ;62
                01  04FD 11809          .BYTE   ^X01
                    04FE 11810          VBUSDATA        <^X1E>,1                ;63
                01  04FF 11811          .BYTE   ^X01
                    0500 11812          VBUSDATA        <^X1E>,1                ;64
                01  0501 11813          .BYTE   ^X01
```

```
                0502 11814          VBUSDATA        <^X1E>,1                ;65
          01    0503 11815          .BYTE   ^X01
                0504 11816          VBUSDATA        <^X1E>,1                ;66
          01    0505 11817          .BYTE   ^X01
                0506 11818          VBUSDATA        <^X1E>,1                ;67
          01    0507 11819          .BYTE   ^X01
                0508 11820          VBUSDATA        <^X1E>,1                ;68
          01    0509 11821          .BYTE   ^X01
                050A 11822          VBUSDATA        <^X1E>,1                ;69
          01    050B 11823          .BYTE   ^X01
                050C 11824          VBUSDATA        <^X1E>,1                ;6A
          01    050D 11825          .BYTE   ^X01
                050E 11826          VBUSDATA        <^X1E>,1                ;6B
          01    050F 11827          .BYTE   ^X01
                0510 11828          VBUSDATA        <^X1E>,1                ;6C
          01    0511 11829          .BYTE   ^X01
                0512 11830          VBUSDATA        <^X1E>,1                ;6D
          01    0513 11831          .BYTE   ^X01
                0514 11832          VBUSDATA        <^X1E>,1                ;6E
          01    0515 11833          .BYTE   ^X01
                0516 11834          VBUSDATA        <^X1E>,1                ;6F
          01    0517 11835          .BYTE   ^X01
                0518 11836          VBUSDATA        <^X1E>,1                ;70
          01    0519 11837          .BYTE   ^X01
                051A 11838          VBUSDATA        <^X1E>,1                ;71
          01    051B 11839          .BYTE   ^X01
                051C 11840          VBUSDATA        <^X1E>,1                ;72
          01    051D 11841          .BYTE   ^X01
                051E 11842          VBUSDATA        <^X1E>,1                ;73
          01    051F 11843          .BYTE   ^X01
                0520 11844          VBUSDATA        <^X1E>,1                ;74
          01    0521 11845          .BYTE   ^X01
                0522 11846          VBUSDATA        <^X1E>,1                ;75
          01    0523 11847          .BYTE   ^X01
                0524 11848          VBUSDATA        <^X1E>,1                ;76
          01    0525 11849          .BYTE   ^X01
                0526 11850          VBUSDATA        <^X1E>,1                ;77
          01    0527 11851          .BYTE   ^X01
                0528 11852          VBUSDATA        <^X1E>,1                ;78
          01    0529 11853          .BYTE   ^X01
                052A 11854          VBUSDATA        <^X1E>,1                ;79
          01    052B 11855          .BYTE   ^X01
                052C 11856          VBUSDATA        <^X1E>,1                ;7A
          01    052D 11857          .BYTE   ^X01
                052E 11858          VBUSDATA        <^X1E>,1                ;7B
          01    052F 11859          .BYTE   ^X01
                0530 11860          VBUSDATA        <^X1E>,1                ;7C
          01    0531 11861          .BYTE   ^X01
                0532 11862          VBUSDATA        <^X1E>,1                ;7D
          01    0533 11863          .BYTE   ^X01
                0534 11864          VBUSDATA        <^X1E>,1                ;7E
          01    0535 11865          .BYTE   ^X01
                0536 11866          VBUSDATA        <^X1E>,1                ;7F
          01    0537 11867          .BYTE   ^X01
                0538 11868          VBUSDATA        <^X1E>,1                ;80
```

```
01  0539 11869          .BYTE   ^X01
    053A 11870          VBUSDATA        <^X1E>,0                ;81
01  053B 11871          .BYTE   ^X01
    053C 11872          VBUSDATA        <^X1E>,1                ;82
01  053D 11873          .BYTE   ^X01
    053E 11874          VBUSDATA        <^X1E>,0                ;83
01  053F 11875          .BYTE   ^X01
    0540 11876          VBUSDATA        <^X1E>,1                ;84
01  0541 11877          .BYTE   ^X01
    0542 11878          VBUSDATA        <^X1E>,1                ;85
01  0543 11879          .BYTE   ^X01
    0544 11880          VBUSDATA        <^X1E>,1                ;86
01  0545 11881          .BYTE   ^X01
    0546 11882          VBUSDATA        <^X1E>,1                ;87
01  0547 11883          .BYTE   ^X01
    0548 11884          VBUSDATA        <^X1E>,1                ;88
01  0549 11885          .BYTE   ^X01
    054A 11886          VBUSDATA        <^X1E>,0                ;89
01  054B 11887          .BYTE   ^X01
    054C 11888          VBUSDATA        <^X1E>,1                ;8A
01  054D 11889          .BYTE   ^X01
    054E 11890          VBUSDATA        <^X1E>,0                ;8B
01  054F 11891          .BYTE   ^X01
    0550 11892          VBUSDATA        <^X1E>,1                ;8C
01  0551 11893          .BYTE   ^X01
    0552 11894          VBUSDATA        <^X1E>,0                ;8D
01  0553 11895          .BYTE   ^X01
    0554 11896          VBUSDATA        <^X1E>,1                ;8E
01  0555 11897          .BYTE   ^X01
    0556 11898          VBUSDATA        <^X1E>,0                ;8F
01  0557 11899          .BYTE   ^X01
    0558 11900          VBUSDATA        <^X1E>,1                ;90
01  0559 11901          .BYTE   ^X01
    055A 11902          VBUSDATA        <^X1E>,1                ;91
01  055B 11903          .BYTE   ^X01
    055C 11904          VBUSDATA        <^X1E>,1                ;92
01  055D 11905          .BYTE   ^X01
    055E 11906          VBUSDATA        <^X1E>,1                ;93
01  055F 11907          .BYTE   ^X01
    0560 11908          VBUSDATA        <^X1E>,1                ;94
01  0561 11909          .BYTE   ^X01
    0562 11910          VBUSDATA        <^X1E>,1                ;95
01  0563 11911          .BYTE   ^X01
    0564 11912          VBUSDATA        <^X1E>,1                ;96
01  0565 11913          .BYTE   ^X01
    0566 11914          VBUSDATA        <^X1E>,1                ;97
01  0567 11915          .BYTE   ^X01
    0568 11916          VBUSDATA        <^X1E>,1                ;98
01  0569 11917          .BYTE   ^X01
    056A 11918          VBUSDATA        <^X1E>,1                ;99
01  056B 11919          .BYTE   ^X01
    056C 11920          VBUSDATA        <^X1E>,1                ;9A
01  056D 11921          .BYTE   ^X01
    056E 11922          VBUSDATA        <^X1E>,1                ;9B
01  056F 11923          .BYTE   ^X01
```

```
                0570 11924          VBUSDATA        <^X1E>,0                ;9C
01              0571 11925          .BYTE   ^X01
                0572 11926          VBUSDATA        <^X1E>,0                ;9D
01              0573 11927          .BYTE   ^X01
                0574 11928          VBUSDATA        <^X1E>,1                ;9E
01              0575 11929          .BYTE   ^X01
                0576 11930          VBUSDATA        <^X1E>,1                ;9F
01              0577 11931          .BYTE   ^X01
                0578 11932          VBUSDATA        <^X1E>,1                ;A0
01              0579 11933          .BYTE   ^X01
                057A 11934          VBUSDATA        <^X1E>,0                ;A1
01              057B 11935          .BYTE   ^X01
                057C 11936          VBUSDATA        <^X1E>,1                ;A2
01              057D 11937          .BYTE   ^X01
                057E 11938          VBUSDATA        <^X1E>,0                ;A3
01              057F 11939          .BYTE   ^X01
                0580 11940          VBUSDATA        <^X1E>,1                ;A4
01              0581 11941          .BYTE   ^X01
                0582 11942          VBUSDATA        <^X1E>,1                ;A5
01              0583 11943          .BYTE   ^X01
                0584 11944          VBUSDATA        <^X1E>,1                ;A6
01              0585 11945          .BYTE   ^X01
                0586 11946          VBUSDATA        <^X1E>,1                ;A7
01              0587 11947          .BYTE   ^X01
                0588 11948          VBUSDATA        <^X1E>,1                ;A8
01              0589 11949          .BYTE   ^X01
                058A 11950          VBUSDATA        <^X1E>,0                ;A9
01              058B 11951          .BYTE   ^X01
                058C 11952          VBUSDATA        <^X1E>,1                ;AA
01              058D 11953          .BYTE   ^X01
                058E 11954          VBUSDATA        <^X1E>,0                ;AB
01              058F 11955          .BYTE   ^X01
                0590 11956          VBUSDATA        <^X1E>,1                ;AC
01              0591 11957          .BYTE   ^X01
                0592 11958          VBUSDATA        <^X1E>,0                ;AD
01              0593 11959          .BYTE   ^X01
                0594 11960          VBUSDATA        <^X1E>,1                ;AE
01              0595 11961          .BYTE   ^X01
                0596 11962          VBUSDATA        <^X1E>,0                ;AF
01              0597 11963          .BYTE   ^X01
                0598 11964          VBUSDATA        <^X1E>,1                ;B0
01              0599 11965          .BYTE   ^X01
                059A 11966          VBUSDATA        <^X1E>,1                ;B1
01              059B 11967          .BYTE   ^X01
                059C 11968          VBUSDATA        <^X1E>,1                ;B2
01              059D 11969          .BYTE   ^X01
                059E 11970          VBUSDATA        <^X1E>,1                ;B3
01              059F 11971          .BYTE   ^X01
                05A0 11972          VBUSDATA        <^X1E>,1                ;B4
01              05A1 11973          .BYTE   ^X01
                05A2 11974          VBUSDATA        <^X1E>,1                ;B5
01              05A3 11975          .BYTE   ^X01
                05A4 11976          VBUSDATA        <^X1E>,1                ;B6
01              05A5 11977          .BYTE   ^X01
                05A6 11978          VBUSDATA        <^X1E>,1                ;B7
```

```
01  05A7 11979          .BYTE   ^X01
    05A8 11980          VBUSDATA        <^X1E>,1                ;B8
01  05A9 11981          .BYTE   ^X01
    05AA 11982          VBUSDATA        <^X1E>,1                ;B9
01  05AB 11983          .BYTE   ^X01
    05AC 11984          VBUSDATA        <^X1E>,1                ;BA
01  05AD 11985          .BYTE   ^X01
    05AE 11986          VBUSDATA        <^X1E>,1                ;BB
01  05AF 11987          .BYTE   ^X01
    05B0 11988          VBUSDATA        <^X1E>,1                ;BC
01  05B1 11989          .BYTE   ^X01
    05B2 11990          VBUSDATA        <^X1E>,1                ;BD
01  05B3 11991          .BYTE   ^X01
    05B4 11992          VBUSDATA        <^X1E>,1                ;BE
01  05B5 11993          .BYTE   ^X01
    05B6 11994          VBUSDATA        <^X1E>,1                ;BF
01  05B7 11995          .BYTE   ^X01
    05B8 11996          VBUSDATA        <^X1E>,1                ;C0
01  05B9 11997          .BYTE   ^X01
    05BA 11998          VBUSDATA        <^X1E>,0                ;C1
01  05BB 11999          .BYTE   ^X01
    05BC 12000          VBUSDATA        <^X1E>,1                ;C2
01  05BD 12001          .BYTE   ^X01
    05BE 12002          VBUSDATA        <^X1E>,0                ;C3
01  05BF 12003          .BYTE   ^X01
    05C0 12004          VBUSDATA        <^X1E>,1                ;C4
01  05C1 12005          .BYTE   ^X01
    05C2 12006          VBUSDATA        <^X1E>,1                ;C5
01  05C3 12007          .BYTE   ^X01
    05C4 12008          VBUSDATA        <^X1E>,1                ;C6
01  05C5 12009          .BYTE   ^X01
    05C6 12010          VBUSDATA        <^X1E>,1                ;C7
01  05C7 12011          .BYTE   ^X01
    05C8 12012          VBUSDATA        <^X1E>,1                ;C8
01  05C9 12013          .BYTE   ^X01
    05CA 12014          VBUSDATA        <^X1E>,0                ;C9
01  05CB 12015          .BYTE   ^X01
    05CC 12016          VBUSDATA        <^X1E>,1                ;CA
01  05CD 12017          .BYTE   ^X01
    05CE 12018          VBUSDATA        <^X1E>,0                ;CB
01  05CF 12019          .BYTE   ^X01
    05D0 12020          VBUSDATA        <^X1E>,1                ;CC
01  05D1 12021          .BYTE   ^X01
    05D2 12022          VBUSDATA        <^X1E>,0                ;CD
01  05D3 12023          .BYTE   ^X01
    05D4 12024          VBUSDATA        <^X1E>,1                ;CE
01  05D5 12025          .BYTE   ^X01
    05D6 12026          VBUSDATA        <^X1E>,0                ;CF
01  05D7 12027          .BYTE   ^X01
    05D8 12028          VBUSDATA        <^X1E>,1                ;D0
01  05D9 12029          .BYTE   ^X01
    05DA 12030          VBUSDATA        <^X1E>,1                ;D1
01  05DB 12031          .BYTE   ^X01
    05DC 12032          VBUSDATA        <^X1E>,1                ;D2
01  05DD 12033          .BYTE   ^X01
```

```
                05DE 12034          VBUSDATA        <^X1E>,1                ;D3
             01 05DF 12035          .BYTE   ^X01
                05E0 12036          VBUSDATA        <^X1E>,1                ;D4
             01 05E1 12037          .BYTE   ^X01
                05E2 12038          VBUSDATA        <^X1E>,1                ;D5
             01 05E3 12039          .BYTE   ^X01
                05E4 12040          VBUSDATA        <^X1E>,1                ;D6
             01 05E5 12041          .BYTE   ^X01
                05E6 12042          VBUSDATA        <^X1E>,1                ;D7
             01 05E7 12043          .BYTE   ^X01
                05E8 12044          VBUSDATA        <^X1E>,1                ;D8
             01 05E9 12045          .BYTE   ^X01
                05EA 12046          VBUSDATA        <^X1E>,1                ;D9
             01 05EB 12047          .BYTE   ^X01
                05EC 12048          VBUSDATA        <^X1E>,1                ;DA
             01 05ED 12049          .BYTE   ^X01
                05EE 12050          VBUSDATA        <^X1E>,1                ;DB
             01 05EF 12051          .BYTE   ^X01
                05F0 12052          VBUSDATA        <^X1E>,1                ;DC
             01 05F1 12053          .BYTE   ^X01
                05F2 12054          VBUSDATA        <^X1E>,1                ;DD
             01 05F3 12055          .BYTE   ^X01
                05F4 12056          VBUSDATA        <^X1E>,1                ;DE
             01 05F5 12057          .BYTE   ^X01
                05F6 12058          VBUSDATA        <^X1E>,1                ;DF
             01 05F7 12059          .BYTE   ^X01
                05F8 12060          VBUSDATA        <^X1E>,1                ;E0
             01 05F9 12061          .BYTE   ^X01
                05FA 12062          VBUSDATA        <^X1E>,1                ;E1
             01 05FB 12063          .BYTE   ^X01
                05FC 12064          VBUSDATA        <^X1E>,1                ;E2
             01 05FD 12065          .BYTE   ^X01
                05FE 12066          VBUSDATA        <^X1E>,1                ;E3
             01 05FF 12067          .BYTE   ^X01
                0600 12068          VBUSDATA        <^X1E>,1                ;E4
             01 0601 12069          .BYTE   ^X01
                0602 12070          VBUSDATA        <^X1E>,1                ;E5
             01 0603 12071          .BYTE   ^X01
                0604 12072          VBUSDATA        <^X1E>,1                ;E6
             01 0605 12073          .BYTE   ^X01
                0606 12074          VBUSDATA        <^X1E>,1                ;E7
             01 0607 12075          .BYTE   ^X01
                0608 12076          VBUSDATA        <^X1E>,1                ;E8
             01 0609 12077          .BYTE   ^X01
                060A 12078          VBUSDATA        <^X1E>,1                ;E9
             01 060B 12079          .BYTE   ^X01
                060C 12080          VBUSDATA        <^X1E>,0                ;EA
             01 060D 12081          .BYTE   ^X01
                060E 12082          VBUSDATA        <^X1E>,0                ;EB
             01 060F 12083          .BYTE   ^X01
                0610 12084          VBUSDATA        <^X1E>,0                ;EC
             01 0611 12085          .BYTE   ^X01
                0612 12086          VBUSDATA        <^X1E>,0                ;ED
             01 0613 12087          .BYTE   ^X01
                0614 12088          VBUSDATA        <^X1E>,0                ;EE
```

```
                    01  0615 12089          .BYTE   ^X01
                        0616 12090          VBUSDATA        <^X1E>,0                ;EF
                    01  0617 12091          .BYTE   ^X01
                        0618 12092          VBUSDATA        <^X1E>,1                ;F0
                    01  0619 12093          .BYTE   ^X01
                        061A 12094          VBUSDATA        <^X1E>,0                ;F1
                    01  061B 12095          .BYTE   ^X01
                        061C 12096          VBUSDATA        <^X1E>,1                ;F2
                    01  061D 12097          .BYTE   ^X01
                        061E 12098          VBUSDATA        <^X1E>,1                ;F3
                    01  061F 12099          .BYTE   ^X01
                        0620 12100          VBUSDATA        <^X1E>,1                ;F4
                    01  0621 12101          .BYTE   ^X01
                        0622 12102          VBUSDATA        <^X1E>,1                ;F5
                    01  0623 12103          .BYTE   ^X01
                        0624 12104          VBUSDATA        <^X1E>,1                ;F6
                    01  0625 12105          .BYTE   ^X01
                        0626 12106          VBUSDATA        <^X1E>,1                ;F7
                    01  0627 12107          .BYTE   ^X01
                        0628 12108          VBUSDATA        <^X1E>,1                ;F8
                    01  0629 12109          .BYTE   ^X01
                        062A 12110          VBUSDATA        <^X1E>,1                ;F9
                    01  062B 12111          .BYTE   ^X01
                        062C 12112          VBUSDATA        <^X1E>,1                ;FA
                    01  062D 12113          .BYTE   ^X01
                        062E 12114          VBUSDATA        <^X1E>,1                ;FB
                    01  062F 12115          .BYTE   ^X01
                        0630 12116          VBUSDATA        <^X1E>,1                ;FC
                    01  0631 12117          .BYTE   ^X01
                        0632 12118          VBUSDATA        <^X1E>,1                ;FD
                    01  0633 12119          .BYTE   ^X01
                        0634 12120          VBUSDATA        <^X1E>,1                ;FE
                    01  0635 12121          .BYTE   ^X01
                        0636 12122          VBUSDATA        <^X1E>,1                ;FF
                        0637 12123
                        0637 12124 BEGIN_CPU_DATA
                        0002 12125
                        0002 12126 DCS_DATA
                        0001 12127
U 00, 7700,C800,0364,0300,0470,1801     0001 12128 ;% 00    NOP                     ; get cpu up to speed
U 01, 5700,C840,0364,0308,0470,1802     000C 12132 ;% 01    R[TEMP2]_RDM            ; load opcode in R[TEMP2]
U 02, 7700,F800,4ECF,4FFF,8470,1803     0017 12136 ;% 02    LONLIT_[62616000]       ; load opspecs in lonlit reg.
U 03, 7700,8800,5BE5,03D4,0470,1804     0022 12140 ;% 03    Q_R[LONLIT]             ; load opspecs in Q register
U 04, 7700,0842,00A4,0308,0470,1805     002D 12144 ;% 04    R[TEMP2]_M[TEMP2].OR.Q  ; OR opcode with opspecs
U 05, 7700,4868,5BE4,0308,0470,1806     0038 12148 ;% 05    M[TEMP8]_R[TEMP2]       ; save in M[TEMP8] for error
U 06, 7700,C800,5BE4,03D8,4A70,1807     0043 12152 ;% 06    VA_R[ZERO]              ; zero the VA register
U 07, 7701,C800,5BE4,0208,5590,1808     004E 12156 ;% 07    WRITE.LONG R[TEMP2]     ; write data for XB0
U 08, 7700,0800,5BE4,0308,5470,1809     0059 12160 ;% 08    WDR_UNROT(R[TEMP2])     ; validate cache location 0
U 09, 7700,C800,0364,0300,4470,180A     0064 12164 ;% 09    VA_VA+4                 ; update VA by 4
U 0A, 7700,3800,4CCD,4DCE,0470,180B     006F 12168 ;% 0A    LONLIT_[66656463]       ; load opspecs in lonlit reg.
U 0B, 7700,4869,5BE4,03D4,0470,180C     007A 12172 ;% 0B    M[TEMP9]_R[LONLIT]      ; save in M[TEMP9] for error
U 0C, 7701,8800,5BE4,02D4,5590,180D     0085 12176 ;% 0C    WRITE.LONG R[LONLIT]    ; write data for XB1
U 0D, 7700,C800,5BE4,03D4,5470,180E     0090 12180 ;% 0D    WDR_UNROT(R[LONLIT])    ; validate cache location 4
U 0E, 7700,C800,5BE4,03D8,4870,180F     009B 12184 ;% 0E    PC_R[ZERO]              ; zero PC to force a prefetch
U 0F, 7700,C800,0364,1700,0470,1810     00A6 12188 ;% 0F    BUT/IRD1TST             ; perform ird1
```

```
U 10, 7300,4800,0364,0300,0470,1811  00B1 12192 ;% 10   NOP,STOP.CLOCK                          ; halt
                                     00BC 12196
U 11, 7700,4800,0364,0300,0470,1812  00BC 12197 ;% 11   NOP                                   ; get cpu up to speed
U 12, 7700,4800,0364,1B00,0470,1800  00C7 12201 ;% 12   BUT/LOD.INC.BRA,NEXT/1800             ; load OSR & increment IRD cntr
U 13, 7300,4800,0364,0300,0470,1814  00D2 12205 ;% 13   STOP.CLOCK                           ; halt
                                     00DD 12209
U 14, 7700,C800,0364,0300,0470,1815  00DD 12210 ;% 14   NOP                                   ; get cpu up to speed
U 15, 5700,C800,0364,0300,2A70,1816  00E8 12214 ;% 15   WCTRL/FPA.ENABLE_WB5,WB_RDM           ; write the FPA enable bit
U 16, 7700,4800,0364,0300,0470,1817  00F3 12218 ;% 16   NOP                                   ; ...
U 17, 7700,4800,0364,C300,0470,1800  00FE 12222 ;% 17   BUT/NO.FPA,NEXT/1800                  ; check FPA status
U 18, 7300,C800,0364,0300,0470,1819  0109 12226 ;% 18   STOP.CLOCK                           ; halt
                                     0114 12230
U 19, 7700,C800,0364,0300,0470,181A  0114 12231 ;% 19   NOP                                   ; get the cpu up to speed
U 1A, 7700,F800,7F9F,FFFF,8470,181B  011F 12235 ;% 1A   LONLIT_[00C00000]                     ; address of s&c bit in crar
U 1B, 7700,4800,5BE4,03D4,2470,181C  012A 12239 ;% 1B   WCTRL/LOADCRAR,WB_R[LONLIT]          ; address the s&c bit
U 1C, 7700,B800,7FDF,FFFF,8470,181D  0135 12243 ;% 1C   LONLIT_[00400000]                     ; data for ctrl-p int. disable
U 1D, 7700,4800,5BE4,03D4,2C70,181E  0140 12247 ;% 1D   WCTRL/CONWRITE,WB_R[LONLIT]          ; disable ctrl-p interrupts
U 1E, 7300,C800,0364,0300,0470,181F  014B 12251 ;% 1E   STOP.CLOCK                           ; halt
                                     0156 12255
U 1F, 7700,C800,0364,0300,0470,1820  0156 12256 ;% 1F   NOP                                   ; get the cpu up to speed
U 20, 7700,F800,7F9F,FFFF,8470,1821  0161 12260 ;% 20   LONLIT_[00C00000]                     ; address of s&c bit in crar
U 21, 7700,C800,5BE4,03D4,2470,1822  016C 12264 ;% 21   WCTRL/LOADCRAR,WB_R[LONLIT]          ; address the s&c bit
U 22, 7700,7800,7FB7,FFFF,8470,1823  0177 12268 ;% 22   LONLIT_[00900000]                     ; data for ctrl-p int. enable
U 23, 7700,4800,5BE4,03D4,2C70,1824  0182 12272 ;% 23   WCTRL/CONWRITE,WB_R[LONLIT]          ; enable ctrl-p interrupts
U 24, 7300,C800,0364,0300,0470,1825  018D 12276 ;% 24   STOP.CLOCK                           ; halt
                                     0198 12280
                                     0198 12281 ; Set-up RTEMP0 and RTEMP1 for error callout
                                     0198 12282
U 25, 7700,C800,0364,0300,0470,1826  0198 12283 ;% 25   NOP                                   ; get cpu up to speed
U 26, 7700,0848,5924,0300,0470,1827  01A3 12287 ;% 26   R[TEMP0]_M[TEMP8]                     ; move XB0 data to R[TEMP0]
U 27, 7700,0849,5924,0304,0470,1828  01AE 12291 ;% 27   R[TEMP1]_M[TEMP9]                     ; move XB1 data to R[TEMP1]
U 28, 7300,C800,0364,0300,0470,1829  01B9 12295 ;% 28   STOP.CLOCK                           ; halt
                                     01C4 12299
                                     01C4 12300 RTEMP_DATA
                                     0001 12301
                           00000000  0001 12302         .LONG   0                          ; rtemp0 = contents of xb0
                           00000000  0005 12303         .LONG   0                          ; rtemp1 = contents of xb1
                           00000000  0009 12304         .LONG   0                          ; rtemp2 = instruction
                                     000D 12305
                                     000D 12306 END_CPU_DATA
                                     0637 12307
                                     0637 12308 END_TEST_DATA
                                     0637
                                     0637       ;-----------------------------------------------------------------------
                                     0637 12309
                                     0637 12310 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
                        0032        .SBTTL  "TEST   0CC     Second IRDX Instruction Address Decode Test 3
                        0032 12312 ;*******************************************************************************
                        0032 12313 ;++
                        0032 12314 ;
                        0032 12315 ; FUNCTIONAL DESCRIPTION
                        0032 12316 ;
                        0032 12317 ;       This test will examine the Native Mode IRDX roms when the IRD Counter
                        0032 12318 ;       is equal to one and the operand specifier is for a register reference.
                        0032 12319 ;
                        0032 12320 ; TEST ALGORITHM
                        0032 12321 ;
                        0032 12322 ;       This test consists of three subtests.  The first subtest will examine
                        0032 12323 ;       the IRDX roms when the Floating Point Accelerator (FPA) is not present
                        0032 12324 ;       in the system.  The second subtest will examine the IRDX roms when the
                        0032 12325 ;       FPA is present but disabled.  The third subtest will examine the IRDX
                        0032 12326 ;       roms when the FPA is present and enabled.  The algorithm is as follows:
                        0032 12327 ;
                        0032 12328 ;        1.  Initialize the processor.
                        0032 12329 ;        2.  Disable CTRL/P interrupts.
                        0032 12330 ;        3.  Check to see if FPA present
                        0032 12331 ;               If FPA present
                        0032 12332 ;                   Then skip to Subtest 2
                        0032 12333 ;               Else fall through to subtest 1
                        0032 12334 ;
                        0032 12335 ;       Subtest 1
                        0032 12336 ;
                        0032 12337 ;        4.  Set-up flag to expect a micro trap.
                        0032 12338 ;        5.  Initialize loop I for 256 opcodes (00-FF).
                        0032 12339 ;        6.  Store the index I in a location that will be used to load into the
                        0032 12340 ;            WDREG.  The value is used as the opcode.
                        0032 12341 ;        7.  Loop on error from here.
                        0032 12342 ;        8.  Set-up DCS location 12 with a BUT/LOD.INC.BRA?,NEXT/1800.  This
                        0032 12343 ;            is used to get the IRD counter to 1 without sourcing an address
                        0032 12344 ;            from the IRDX roms (ie., by doing an IRDX).
                        0032 12345 ;        9.  Execute the following test microcode (performs the IRD1):
                        0032 12346 ;               a.  NOP to get the RDM up to speed.
                        0032 12347 ;               b.  Read the WDREG into RTEMP2 (opcode).
                        0032 12348 ;               c.  Load the operands into the LONLIT register.
                        0032 12349 ;               d.  Store the operands into the Q register to prepare for ORing
                        0032 12350 ;                   the opcode with the operands.
                        0032 12351 ;               e.  OR the opcode and operands together (XB0 data).
                        0032 12352 ;               f.  Store XB0 data in M[TEMP8] in case needed for error callout.
                        0032 12353 ;               g.  Zero the VA.
                        0032 12354 ;               h.  Validate cache location 0 by first writing the data in
                        0032 12355 ;                   RTEMP2 then reading the data back.
                        0032 12356 ;               i.  Update the VA by 4.
                        0032 12357 ;               j.  Load the second operand data into the LONLIT register
                        0032 12358 ;                   (XB1 data).
                        0032 12359 ;               k.  Store XB1 data in M[TEMP9] in case needed for error callout.
                        0032 12360 ;               l.  Validate cache location 4 by first writing the data in
                        0032 12361 ;                   the LONLIT register then reading the data back.
                        0032 12362 ;               m.  Load the PC with zero to cause a prefetch to take place.
                        0032 12363 ;               n.  Do an IRD1 to init the IRD counter and prepare for instruc-
                        0032 12364 ;                   tion/operand specifier evaluation.
                        0032 12365 ;               o.  Halt.
```

```
                        0032 12366 ;       10.  Perform the following test microcode (performs the LOD.INC.BRA?)
                        0032 12367 ;              a.  NOP to get the RDM up to speed.
                        0032 12368 ;              b.  Do an BUT/LOD.INC.BRA?,NEXT/1800
                        0032 12369 ;              c.  Halt.
                        0032 12370 ;       11.  Load DCS location 12 with a BUT/IRDX micro order.
                        0032 12371 ;       12.  Perform the following test microcode (performs the IRDX with the
                        0032 12372 ;            IRD counter = 1):
                        0032 12373 ;              a.  NOP to get the RDM up to speed.
                        0032 12374 ;              b.  Do an IRDX.
                        0032 12375 ;              c.  Halt.
                        0032 12376 ;       13.  Compare the contents of the CSTRP register (output from the IRDX
                        0032 12377 ;            Roms) against the Rom Table.
                        0032 12378 ;       14.  If the two do not equal, execute the following microcode and
                        0032 12379 ;            signal an error (this is done to set-up for printing RTEMP0 & 1
                        0032 12380 ;            on the error callout):
                        0032 12381 ;              a.  NOP to get the RDM up to speed.
                        0032 12382 ;              b.  Load RTEMP0 with the contents of M[TEMP8]
                        0032 12383 ;              c.  Load RTEMP1 with the contents of M[TEMP9]
                        0032 12384 ;              d.  Halt
                        0032 12385 ;       15.  Continue with loop I for the 256 VAX opcodes.
                        0032 12386 ;       16.  Re-enable CTRL/P interrupts then skip to next tests.
                        0032 12387 ;
                        0032 12388 ;       Subtest 2
                        0032 12389 ;
                        0032 12390 ;       17.  Load the FPA disable bit (bit <5> equal to 0).
                        0032 12391 ;       18.  Loop on error from here.
                        0032 12392 ;       19.  Set-up to expect micro trap.
                        0032 12393 ;       20.  Execute the test microcode.  This is done only to get the FPA to
                        0032 12394 ;            a state where it can be disabled or enabled.  If this had not been
                        0032 12395 ;            done before then the FPA would be setting in the initialize state
                        0032 12396 ;            which forces FPA enable.
                        0032 12397 ;       21.  Execute the following microcode to disable the FPA:
                        0032 12398 ;              a.  NOP to get the RDM up to speed.
                        0032 12399 ;              b.  Read the RDM WDREG onto the W Bus and then write the W Bus
                        0032 12400 ;                  to the FPA.
                        0032 12401 ;              c.  Perform a NO.FPA test to insure that the FPA got disabled.
                        0032 12402 ;              d.  Halt.
                        0032 12403 ;       22.  Check the Trap register to insure that the FPA got disabled.
                        0032 12404 ;       23.  If the FPA is not disabled, then signal an error.
                        0032 12405 ;       24.  Set-up flag to expect a micro trap.
                        0032 12406 ;       25.  Initialize loop I for 256 opcodes (00-FF).
                        0032 12407 ;       26.  Store the index I in a location that will be used to load into the
                        0032 12408 ;            WDREG.  The value is used as the opcode.
                        0032 12409 ;       27.  Loop on error from here.
                        0032 12410 ;       28.  Set-up DCS location 12 with a BUT/LOD.INC.BRA?,NEXT/1800.  This
                        0032 12411 ;            is used to get the IRD counter to 1 without sourcing an address
                        0032 12412 ;            from the IRDX roms (ie., by doing an IRDX).
                        0032 12413 ;       29.  Execute the following test microcode:
                        0032 12414 ;              a.  NOP to get the RDM up to speed.
                        0032 12415 ;              b.  Read the WDREG into RTEMP2 (opcode).
                        0032 12416 ;              c.  Load the operands into the LONLIT register.
                        0032 12417 ;              d.  Store the operands into the Q register to prepare for ORing
                        0032 12418 ;                  the opcode with the operands.
                        0032 12419 ;              e.  OR the opcode and operands together (XB0 data).
                        0032 12420 ;              f.  Store XB0 data in M[TEMP8] in case needed for error callout.
```

```
0032 12421 ;                    g.  Zero the VA.
0032 12422 ;                    h.  Validate cache location 0 by first writing the data in
0032 12423 ;                        RTEMP2 then reading the data back.
0032 12424 ;                    i.  Update the VA by 4.
0032 12425 ;                    j.  Load the second operand data into the LONLIT register
0032 12426 ;                        (XB1 data).
0032 12427 ;                    k.  Store XB1 data in M[TEMP9] in case needed for error callout.
0032 12428 ;                    l.  Validate cache location 4 by first writing the data in
0032 12429 ;                        the LONLIT register then reading the data back.
0032 12430 ;                    m.  Load the PC with zero to cause a prefetch to take place.
0032 12431 ;                    n.  Do an IRD1 to init the IRD counter and prepare for instruc-
0032 12432 ;                        tion/operand specifier evaluation.
0032 12433 ;                    o.  Halt.
0032 12434 ;            30.  Perform the following test microcode.
0032 12435 ;                    a.  NOP to get the RDM up to speed.
0032 12436 ;                    b.  Do an BUT/LOD.INC.BRA?,NEXT/1800
0032 12437 ;                    c.  Halt.
0032 12438 ;            31.  Load DCS location 12 with a BUT/IRDX micro order.
0032 12439 ;            32.  Perform the following test microcode (performs the IRDX with the
0032 12440 ;                 IRD counter = 1):
0032 12441 ;                    a.  NOP to get the RDM up to speed.
0032 12442 ;                    b.  Do an IRDX.
0032 12443 ;                    c.  Halt.
0032 12444 ;            33.  Compare the contents of the CSTRP register (output from the IRDX
0032 12445 ;                 Roms) against the Rom Table.
0032 12446 ;            34.  If the two do not equal, execute the following microcode and
0032 12447 ;                 signal an error (this is done to set-up for printing RTEMP0 & 1
0032 12448 ;                 on the error callout):
0032 12449 ;                    a.  NOP to get the RDM up to speed.
0032 12450 ;                    b.  Load RTEMP0 with the contents of M[TEMP8]
0032 12451 ;                    c.  Load RTEMP1 with the contents of M[TEMP9]
0032 12452 ;                    d.  Halt
0032 12453 ;            35.  Continue with loop I for the 256 VAX opcodes.
0032 12454 ;            36.  Go to Subtest 3.
0032 12455 ;
0032 12456 ;            Subtest 3
0032 12457 ;
0032 12458 ;            37.  Load the FPA enable bit (bit <5> equal to 1).
0032 12459 ;            38.  Loop on error from here.
0032 12460 ;            39.  Set-up to expect micro trap.
0032 12461 ;            40.  Execute the following microcode to enable the FPA:
0032 12462 ;                    a.  NOP to get the RDM up to speed.
0032 12463 ;                    b.  Read the RDM WDREG onto the W Bus and then write the W Bus
0032 12464 ;                        to the FPA.
0032 12465 ;                    c.  Perform a NO.FPA test to insure that the FPA got enabled.
0032 12466 ;                    d.  Halt.
0032 12467 ;            41.  Check the Trap register to insure that the FPA got enabled.
0032 12468 ;            42.  If the FPA is not enabled, then signal an error.
0032 12469 ;            43.  Set-up flag to expect a micro trap.
0032 12470 ;            44.  Initialize loop I for 256 opcodes (00-FF).
0032 12471 ;            45.  Store the index I in a location that will be used to load into the
0032 12472 ;                 WDREG.  The value is used as the opcode.
0032 12473 ;            46.  Loop on error from here.
0032 12474 ;            47.  Set then Clear the DCLO bit in the RDCTRL register to perform an
0032 12475 ;                 initialize.  This is done because when the test microcode
```

```
0032 12476 ;            executes, if the opcode was an FP opcode then there is a
0032 12477 ;            possibility that the processor could be setting there in an
0032 12478 ;            FPAWAIT state (M clock and D clock stalled).  If this were to
0032 12479 ;            happen then the cpu clocks would no longer run and the remaining
0032 12480 ;            test opcodes would fail.
0032 12481 ;       48.  Set-up DCS location 12 with a BUT/LOD.INC.BRA?,NEXT/1800.  This
0032 12482 ;            is used to get the IRD counter to 1 without sourcing an address
0032 12483 ;            from the IRDX roms (ie., by doing an IRDX).
0032 12484 ;       49.  Execute the following test microcode:
0032 12485 ;               a.  NOP to get the RDM up to speed.
0032 12486 ;               b.  Read the WDREG into RTEMP2 (opcode).
0032 12487 ;               c.  Load the operands into the LONLIT register.
0032 12488 ;               d.  Store the operands into the Q register to prepare for ORing
0032 12489 ;                   the opcode with the operands.
0032 12490 ;               e.  OR the opcode and operands together (XB0 data).
0032 12491 ;               f.  Store XB0 data in M[TEMP8] in case needed for error callout.
0032 12492 ;               g.  Zero the VA.
0032 12493 ;               h.  Validate cache location 0 by first writing the data in
0032 12494 ;                   RTEMP2 then reading the data back.
0032 12495 ;               i.  Update the VA by 4.
0032 12496 ;               j.  Load the second operand data into the LONLIT register
0032 12497 ;                   (XB1 data).
0032 12498 ;               k.  Store XB1 data in M[TEMP8] in case needed for error callout.
0032 12499 ;               l.  Validate cache location 4 by first writing the data in
0032 12500 ;                   the LONLIT register then reading the data back.
0032 12501 ;               m.  Load the PC with zero to cause a prefetch to take place.
0032 12502 ;               n.  Do an IRD1 to init the IRD counter and prepare for instruc-
0032 12503 ;                   tion/operand specifier evaluation.
0032 12504 ;               o.  Halt.
0032 12505 ;       50.  Perform the following test microcode.
0032 12506 ;               a.  NOP to get the RDM up to speed.
0032 12507 ;               b.  Do an BUT/LOD.INC.BRA?,NEXT/1800
0032 12508 ;               c.  Halt.
0032 12509 ;       51.  Load DCS location 12 with a BUT/IRDX micro order.
0032 12510 ;       52.  Perform the following test microcode (performs the IRDX with the
0032 12511 ;            IRD counter = 1):
0032 12512 ;               a.  NOP to get the RDM up to speed.
0032 12513 ;               b.  Do an IRDX.
0032 12514 ;               c.  Halt.
0032 12515 ;       53.  Compare the contents of the CSTRP register (output from the IRDX
0032 12516 ;            Roms) against the Rom Table.
0032 12517 ;       54.  If the two do not equal, execute the following microcode and
0032 12518 ;            signal an error (this is done to set-up for printing RTEMP0 & 1
0032 12519 ;            on the error callout):
0032 12520 ;               a.  NOP to get the RDM up to speed.
0032 12521 ;               b.  Load RTEMP0 with the contents of M[TEMP8]
0032 12522 ;               c.  Load RTEMP1 with the contents of M[TEMP9]
0032 12523 ;               d.  Halt
0032 12524 ;       55.  Continue with loop I for the 256 VAX opcodes.
0032 12525 ;       56.  Re-enable CTRL/P interrupts.
0032 12526 ;       57.  Skip to next tests.
0032 12527 ;
0032 12528 ; TEST PATTERNS
0032 12529 ;
0032 12530 ;            For this test the opcodes are generated by a simple count pattern.  The
```

```
0032 12531 ;       opcodes are based on the 256 base machine instructions as specified in
0032 12532 ;       the VAX architecture handbook.  The generated opcodes are written to
0032 12533 ;       the CPU and combined (ORed) with three bytes of operand data to form
0032 12534 ;       a machine instruction.  The operands all specify Register Deferred mode
0032 12535 ;       using registers 0-6 and are loaded from the LONLIT register.
0032 12536 ;
0032 12537 ;       Also required are two tables of expected control store addresses for
0032 12538 ;       the second IRDX cycle.  One table contains the expected addresses for
0032 12539 ;       the second IRDX when the FPA is disabled, the other for when it is
0032 12540 ;       enabled.
0032 12541 ;
0032 12542 ; LOGIC DESCRIPTION
0032 12543 ;
0032 12544 ;       This test will check the validity of the IRDX roms on the DPM module.
0032 12545 ;       The IRDX roms are addressed by bits <7:0> of the Instruction Register.
0032 12546 ;       Along with being addressed by the IR, there are three other signals
0032 12547 ;       that affect the address.  They are IRD CTR 0 H, FPA ENABLED L, and
0032 12548 ;       REG MODE H.  This test will examine every 8th(hex) address starting at
0032 12549 ;       address 6 with FPA disabled and starting at address 4 with FPA enabled.
0032 12550 ;
0032 12551 ; ASSUMPTIONS
0032 12552 ;
0032 12553 ;       This test assumes the following logic has already been tested; the
0032 12554 ;       CPU's data path, the cache, the XB, and the IR and OSR.
0032 12555 ;
0032 12556 ; ERROR DESCRIPTION
0032 12557 ;
0032 12558 ;       Subtest 1
0032 12559 ;
0032 12560 ;               Expected CS Address
0032 12561 ;               Received CS Address
0032 12562 ;
0032 12563 ;       Subtest 2
0032 12564 ;
0032 12565 ;               First IFERROR
0032 12566 ;
0032 12567 ;                       Expected CS Address
0032 12568 ;                       Received CS Address
0032 12569 ;
0032 12570 ;               Second IFERROR
0032 12571 ;
0032 12572 ;                       Expected IRDX Rom Contents
0032 12573 ;                       Received IRDX Rom Contents
0032 12574 ;                       Loop I count
0032 12575 ;                       Contents of XB0
0032 12576 ;                       Contents of XB1
0032 12577 ;
0032 12578 ;       Subtest 3
0032 12579 ;
0032 12580 ;               First IFERROR
0032 12581 ;
0032 12582 ;                       Expected CS Address
0032 12583 ;                       Received CS Address
0032 12584 ;
0032 12585 ;               Second IFERROR
```

```
                        0032 12586 ;
                        0032 12587 ;                    Expected IRDX Rom Contents
                        0032 12588 ;                    Received IRDX Rom Contents
                        0032 12589 ;                    Loop I count
                        0032 12590 ;                    Contents of XB0
                        0032 12591 ;                    Contents of XB1
                        0032 12592 ;--
                        0032 12593 ;*****************************************************************************
                        0032 12594
                        0032 12595          INITIALIZE                          ; initialize the comet cpu
                        0034 12596          FETCH           <^X19>              ; disable ctrl-p interrupts
                        0039 12597          BRSTCLK         <^X1E>              ; ...
                        003D 12598          FETCH           <^X16>              ; execute a but/no.fpa to check
                        0042 12599          BRSTCLK         <^X18>              ; for fpa present
                        0046 12600          CMPREGMSK       CSTRP,1$,,2$,,W     ; is the fpa present?
                        0052 12601          SKIP            100$,ONERROR        ; if so then skip to subtest 2
                        0059 12602
                        0059 12603          SUBTEST         1
                        005C
                        005C            ;////////////////////////////////////////////////////////////////////////////
                        005C 12604      ;+
                        005C 12605      ;
                        005C 12606      ; FPA is not present.
                        005C 12607      ;
                        005C 12608      ;-
                        005C 12609          EXPUTRAP                            ; else expect non-dcs addresses
                        005E 12610          LOOP            I,1,256             ; loop for 256 opcodes
                        0065 12611          SAVEINDEX       I,9$                ; save the loop count
                        006A 12612          LOADREG         WDREG,9$,,L         ; load wdreg with loop count
                        0072 12613          ERRLOOP
                        0074 12614          LOADDCS         20$,,<^X12>,1       ; load BUT/LOD.IND.BRA?
                        007B 12615          FETCH           0                   ; execute ird1tst
                        0080 12616          BRSTCLK         <^X10>,INH          ; ...
                        0084 12617          FETCH           <^X11>              ; execute BUT/LOD.INC.BRA?
                        0089 12618          BRSTCLK         <^X13>,INH          ; ...
                        008D 12619          LOADDCS         21$,,<^X12>,1       ; load BUT/IRDX
                        0094 12620          FETCH           <^X11>              ; execute the second irdx
                        0099 12621          BRSTCLK         <^X13>,INH          ; ...
                        009D 12622          CMPREGMSK       CSTRP,30$,I,6$,,W   ; is the cs address correct?
                        00A9 12623          SKIP            50$,NOERROR         ; continue testing if no error
                        00B0 12624          FETCH           <^X25>              ; start dcs program
                        00B5 12625          BRSTCLK         <^X28>              ; end dcs program
                        00B9 12626          IFERROR         2,<<IRD>>,<<MIC>>   ; signal an error
                        00C0 12627 50$:     ENDLOOP         I                   ; continue loop
                        00C3 12628          FETCH           <^X1F>              ; re-enable ctrl-p interrupts
                        00C8 12629          BRSTCLK         <^X24>              ; ...
                        00CC 12630          SKIP                                ; skip on to the next test
                        00D3 12631
                        00D3 12632 100$:    SUBTEST         2
                        00D6
                        00D6            ;////////////////////////////////////////////////////////////////////////////
                        00D6 12633      ;+
                        00D6 12634      ;
                        00D6 12635      ; FPA present but not enabled
                        00D6 12636      ;
```

```
00D6 12637 ;-
00D6 12638          LOADREG         WDREG,7$,,L             ; load fpa disable bit
00DE 12639          ERRLOOP                                 ; loop on error here
00E0 12640          EXPUTRAP                                ; expect micro trap
00E2 12641          FETCH           0                       ; start dcs program
00E7 12642          BRSTCLK         <^X10>,INH              ; end dcs program
00EB 12643          FETCH           <^X14>                  ; execute the disable
00F0 12644          BRSTCLK         <^X18>,INH              ; and read back fpa status
00F4 12645          CMPREGMSK       CSTRP,3$,,5$,,W         ; is the fpa disabled?
0100 12646          IFERROR         ,,<FPA>                 ; if not, signal an error
0106 12647          EXPUTRAP                                ; else expect non-dcs addresses
0108 12648          LOOP            I,1,256                 ; loop for 256 opcodes
010F 12649          SAVEINDEX       I,9$                    ; save the loop count
0114 12650          LOADREG         WDREG,9$,,L             ; load wdreg with loop count
011C 12651          ERRLOOP                                 ; loop on error from here
011E 12652          LOADDCS         20$,,<^X12>,1           ; load BUT/LOD.INC.BRA?
0125 12653          FETCH           0                       ; execute ird1tst
012A 12654          BRSTCLK         <^X10>,INH              ; ...
012E 12655          FETCH           <^X11>                  ; execute BUT/LOD.INC.BRA?
0133 12656          BRSTCLK         <^X13>,INH              ; ...
0137 12657          LOADDCS         21$,,<^X12>,1           ; load BUT/IRDX
013E 12658          FETCH           <^X11>                  ; execute the second irdx
0143 12659          BRSTCLK         <^X13>,INH              ; ...
0147 12660          CMPREGMSK       CSTRP,30$,I,6$,,W       ; is the cs address correct?
0153 12661          SKIP            51$,NOERROR             ; continue testing if no error
015A 12662          FETCH           <^X25>                  ; start dcs program
015F 12663          BRSTCLK         <^X28>                  ; end dcs program
0163 12664          IFERROR         2,<<IRD>>,<<MIC>>       ; signal an error
016A 12665 51$:     ENDLOOP         I                       ; continue loop
016D 12666
016D 12667          SUBTEST         3
0170
0170        ;////////////////////////////////////////////////////////////////////////////////
0170 12668 ;+
0170 12669 ;
0170 12670 ; FPA present and enabled
0170 12671 ;
0170 12672 ;-
0170 12673          LOADREG         WDREG,8$,,L             ; load fpa enable bit
0178 12674          ERRLOOP                                 ; loop on error from here
017A 12675          FETCH           <^X14>                  ; enable the fpa
017F 12676          BRSTCLK         <^X18>,INH              ; and read back fpa enable
0183 12677          CMPREGMSK       CSTRP,4$,,5$,,W         ; is the fpa enabled?
018F 12678          IFERROR         ,,<FPA>                 ; if not, signal an error
0195 12679          EXPUTRAP                                ; else expect non-dcs addresses
0197 12680          LOOP            I,1,256                 ; loop for 256 opcodes
019E 12681          SAVEINDEX       I,9$                    ; save the loop count
01A3 12682          LOADREG         WDREG,9$,,L             ; load wdreg with loop count
01AB 12683          ERRLOOP                                 ; loop on error from here
01AD 12684          LOADDCS         20$,,<^X12>,1           ; load BUT/LOD.INC.BRA?
01B4 12685          LOADREG         RDCTRL,10$,,B           ; assert dclo
01BC 12686          LOADREG         RDCTRL,11$,,B           ; de-assert dclo
01C4 12687          FETCH           0                       ; execute ird1tst
01C9 12688          BRSTCLK         <^X10>,INH              ; ...
01CD 12689          FETCH           <^X11>                  ; execute BUT/LOD.INC.BRA?
```

```
                        01D2 12690          BRSTCLK         <^X13>,INH                      ; ...
                        01D6 12691          LOADDCS         21$,,<^X12>,1                   ; load BUT/IRDX
                        01DD 12692          FETCH           <^X11>                          ; execute the second irdx
                        01E2 12693          BRSTCLK         <^X13>,INH                      ; ...
                        01E6 12694          CMPREGMSK       CSTRP,31$,I,6$,,W               ; is the cs address correct?
                        01F2 12695          SKIP            52$,NOERROR                     ; continue testing if no error
                        01F9 12696          FETCH           <^X25>                          ; start dcs program
                        01FE 12697          BRSTCLK         <^X28>                          ; end dcs program
                        0202 12698          IFERROR         2,<<IRD>>,<<MIC>>               ; signal an error
                        0209 12699 52$:     ENDLOOP         I                               ; continue loop
                        020C 12700          FETCH           <^X1F>                          ; re-enable ctrl-p interrupts
                        0211 12701          BRSTCLK         <^X24>                          ; ...
                        0215 12702          SKIP                                            ; end of test
                        021C 12703
                        021C 12704 BEGIN_TEST_DATA
                        021C
                        021C        ;-------------------------------------------------------------------------
                        021C 12705
                1802    021C 12706 1$:      .WORD   ^X1802
                3FFE    021E 12707 2$:      .WORD   ^X3FFE
                        0220 12708
                1801    0220 12709 3$:      .WORD   ^X1801
                1800    0222 12710 4$:      .WORD   ^X1800
                3FFD    0224 12711 5$:      .WORD   ^X3FFD
                        0226 12712
                3FFF    0226 12713 6$:      .WORD   ^X3FFF
                        0228 12714
            00000000    0228 12715 7$:      .LONG   0
            00000020    022C 12716 8$:      .LONG   ^X00000020
                        0230 12717
            00000000    0230 12718 9$:      .LONG   0
                        0234 12719
                  40    0234 12720 10$:     .BYTE   DC_LOW
                  00    0235 12721 11$:     .BYTE   ^X00
                        0236 12722
                        0236 12723 ; Loadable microcode
                        0236 12724
                        0236 12725 20$:
U 11, 7700,4800,0364,1B00,0470,1800     0236 12726 ;% 11    BUT/LOD.INC.BRA,NEXT/1800       ; load OSR & increment IRD cntr
                        0241 12730 21$:
U 11, 7700,0800,0364,0700,0470,1812     0241 12731 ;% 11    BUT/IRDX                        ; do irdx
                        024C 12735
                        024C 12736 ; Expected CS Addresses for Subtests 1 and 2
                        024C 12737 ;
                        024C 12738 ; The following is a table of all the expected control store addresses for the
                        024C 12739 ; second IRDX.
                        024C 12740
 03F8 03F8 03F8 03F8 03F8 03F8 03F8     024C 12741 30$:     .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 00 - 06
 0332 032A 0224 0318 02A8 02A4 03F8     025A 12742          .WORD   ^X03F8,^X02A4,^X02A8,^X0318,^X0224,^X032A,^X0332 ; 07 - 0D
 03F8 03F8 03F8 03F8 03F8 02A2 03AA     0268 12743          .WORD   ^X03AA,^X02A2,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 0E - 14
 03F8 03F8 03F8 03F8 03F8 03F8 03F8     0276 12744          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 15 - 1B
 027C 0294 027C 03F8 03F8 03F8 03F8     0284 12745          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X027C,^X0294,^X027C ; 1C - 22
 0302 0302 0294 0368 0294 0360 0294     0292 12746          .WORD   ^X0294,^X0360,^X0294,^X0368,^X0294,^X0302,^X0302 ; 23 - 29
 03F8 03B3 03B3 03B1 03B1 0274 0274     02A0 12747          .WORD   ^X0274,^X0274,^X03B1,^X03B1,^X03B3,^X03B3,^X03F8 ; 2A - 30
 027C 03BB 0312 0312 00AE 0051 03F8     02AE 12748          .WORD   ^X03F8,^X0051,^X00AE,^X0312,^X0312,^X03BB,^X027C ; 31 - 37
```

```
00EE 039F 00EE 030A 030A 0320 02BC  02BC 12749          .WORD   ^X02BC,^X0320,^X030A,^X030A,^X00EE,^X039F,^X00EE ; 38 - 3E
0081 0081 0351 0351 0088 0088 03F8  02CA 12750          .WORD   ^X03F8,^X0088,^X0088,^X0351,^X0351,^X0081,^X0081 ; 3F - 45
0339 00CC 00C6 00C6 00C6 0095 0095  02D8 12751          .WORD   ^X0095,^X0095,^X00C6,^X00C6,^X00C6,^X00CC,^X0339 ; 46 - 4C
03F8 031D 00A6 030D 00E0 00BC 0339  02E6 12752          .WORD   ^X0339,^X00BC,^X00E0,^X030D,^X00A6,^X031D,^X03F8 ; 4D - 53
03F8 03F8 024E 03F8 00AD 01E1 01D2  02F4 12753          .WORD   ^X01D2,^X01E1,^X00AD,^X03F8,^X024E,^X03F8,^X03F8 ; 54 - 5A
00A2 00A2 02CA 02BA 02B2 02AC 03F8  0302 12754          .WORD   ^X03F8,^X02AC,^X02B2,^X02BA,^X02CA,^X00A2,^X00A2 ; 5B - 61
033D 00E8 00E8 00C0 00C0 00A2 00A2  0310 12755          .WORD   ^X00A2,^X00A2,^X00C0,^X00C0,^X00E8,^X00E8,^X033D ; 62 - 68
00A0 0339 0339 0339 033D 033D 033D  031E 12756          .WORD   ^X033D,^X033D,^X033D,^X0339,^X0339,^X0339,^X00A0 ; 69 - 6F
0341 035B 02C0 03F8 0315 0335 0315  032C 12757          .WORD   ^X0315,^X0335,^X0315,^X03F8,^X02C0,^X035B,^X0341 ; 70 - 76
01E5 03F8 0064 0004 0092 0092 03F8  033A 12758          .WORD   ^X03F8,^X0092,^X0092,^X0004,^X0064,^X03F8,^X01E5 ; 77 - 7D
0297 025B 025B 023D 023D 03F8 01EE  0348 12759          .WORD   ^X01EE,^X03F8,^X023D,^X023D,^X025B,^X025B,^X0297 ; 7E - 84
0291 0291 028B 028B 008C 0309 0040  0356 12760          .WORD   ^X0040,^X0309,^X008C,^X028B,^X028B,^X0291,^X0291 ; 85 - 8B
01AF 0052 00EE 03A3 01CF 0279 0279  0364 12761          .WORD   ^X0279,^X0279,^X01CF,^X03A3,^X00EE,^X0052,^X01AF ; 8C - 92
0051 0051 03F8 03F8 03F8 03F8 000A  0372 12762          .WORD   ^X000A,^X03F8,^X03F8,^X03F8,^X03F8,^X0051,^X0051 ; 93 - 99
023D 03F8 00EE 039F 0211 00EE 00EE  0380 12763          .WORD   ^X00EE,^X00EE,^X0211,^X039F,^X00EE,^X03F8,^X023D ; 9A - A0
008C 0309 0040 0297 025B 025B 023D  038E 12764          .WORD   ^X023D,^X025B,^X025B,^X0297,^X0040,^X0309,^X008C ; A1 - A7
01CF 0279 0279 0291 0291 028B 028B  039C 12765          .WORD   ^X028B,^X028B,^X0291,^X0291,^X0279,^X0279,^X01CF ; A8 - AE
03F8 03F8 000A 01AF 0052 00EE 03A3  03AA 12766          .WORD   ^X03A3,^X00EE,^X0052,^X01AF,^X000A,^X03F8,^X03F8 ; AF - B5
03F8 03F8 03F8 03F8 03F8 03F8 03F8  03B8 12767          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; B6 - BC
025B 025B 023D 023D 03F8 03F8 03F8  03C6 12768          .WORD   ^X03F8,^X03F8,^X03F8,^X023D,^X023D,^X025B,^X025B ; BD - C3
0291 028B 028B 008C 0309 0040 0297  03D4 12769          .WORD   ^X0297,^X0040,^X0309,^X008C,^X028B,^X028B,^X0291 ; C4 - CA
0052 00EE 03A3 01CF 0279 0279 0291  03E2 12770          .WORD   ^X0291,^X0279,^X0279,^X01CF,^X03A3,^X00EE,^X0052 ; CB - D1
0231 03F8 03F8 03F8 03F8 000A 01AF  03F0 12771          .WORD   ^X01AF,^X000A,^X03F8,^X03F8,^X03F8,^X03F8,^X0231 ; D2 - D8
03F8 00EE 03F8 03F8 00EE 0390 0271  03FE 12772          .WORD   ^X0271,^X0390,^X00EE,^X03F8,^X03F8,^X00EE,^X03F8 ; D9 - DF
0287 0287 0287 0287 0287 0287 0287  040C 12773          .WORD   ^X0287,^X0287,^X0287,^X0287,^X0287,^X0287,^X0287 ; E0 - E6
024F 026F 01EF 00EF 03F8 03F8 0287  041A 12774          .WORD   ^X0287,^X03F8,^X03F8,^X00EF,^X01EF,^X026F,^X024F ; E7 - ED
03F8 03A1 03A1 039F 0280 020F 022F  0428 12775          .WORD   ^X022F,^X020F,^X0280,^X039F,^X03A1,^X03A1,^X03F8 ; EE - F4
03A5 03A5 031E 02B4 00AE 00AE 03F8  0436 12776          .WORD   ^X03F8,^X00AE,^X00AE,^X02B4,^X031E,^X03A5,^X03A5 ; F5 - FB
                  03F8 03F8 0776 0776  0444 12777       .WORD   ^X0776,^X0776,^X03F8,^X03F8                      ; FC - FF
                                       044C 12778 ;
                                       044C 12779 ; Expected CS Addresses for Subtest 3
                                       044C 12780 ;
03F8 03F8 03F8 03F8 03F8 03F8 03F8  044C 12781 31$:     .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 00 - 06
0332 032A 0224 0318 02A8 02A4 03F8  045A 12782          .WORD   ^X03F8,^X02A4,^X02A8,^X0318,^X0224,^X032A,^X0332 ; 07 - 0D
03F8 03F8 03F8 03F8 03F8 02A2 03AA  0468 12783          .WORD   ^X03AA,^X02A2,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 0E - 14
03F8 03F8 03F8 03F8 03F8 03F8 03F8  0476 12784          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 15 - 1B
027C 0294 027C 03F8 03F8 03F8 03F8  0484 12785          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X027C,^X0294,^X027C ; 1C - 22
0302 0302 0294 0368 0294 0360 0294  0492 12786          .WORD   ^X0294,^X0360,^X0294,^X0368,^X0294,^X0302,^X0302 ; 23 - 29
03F8 03B3 03B3 03B1 03B1 0274 0274  04A0 12787          .WORD   ^X0274,^X0274,^X03B1,^X03B1,^X03B3,^X03B3,^X03F8 ; 2A - 30
027C 03BB 0312 0312 00AE 0051 03F8  04AE 12788          .WORD   ^X03F8,^X0051,^X00AE,^X0312,^X0312,^X03BB,^X027C ; 31 - 37
00EE 039F 00EE 030A 030A 0320 02BC  04BC 12789          .WORD   ^X02BC,^X0320,^X030A,^X030A,^X00EE,^X039F,^X00EE ; 38 - 3E
00E6 035D 00E6 035D 00E6 035D 03F8  04CA 12790          .WORD   ^X03F8,^X035D,^X00E6,^X035D,^X00E6,^X035D,^X00E6 ; 3F - 45
00E7 00E7 00E7 00E7 00E7 00E6 035D  04D8 12791          .WORD   ^X035D,^X00E6,^X00E7,^X00E7,^X00E7,^X00E7,^X00E7 ; 46 - 4C
03F8 031D 036C 030D 00E0 00E7 00E7  04E6 12792          .WORD   ^X00E7,^X00E7,^X00E0,^X030D,^X036C,^X031D,^X03F8 ; 4D - 53
03F8 03F8 024E 03F8 01EC 0222 00F4  04F4 12793          .WORD   ^X00F4,^X0222,^X01EC,^X03F8,^X024E,^X03F8,^X03F8 ; 54 - 5A
01DA 035E 02CA 02BA 02B2 02AC 03F8  0502 12794          .WORD   ^X03F8,^X02AC,^X02B2,^X02BA,^X02CA,^X035E,^X01DA ; 5B - 61
00E7 01DA 035E 01DA 035E 01DA 035E  0510 12795          .WORD   ^X035E,^X01DA,^X035E,^X01DA,^X035E,^X01DA,^X00E7 ; 62 - 68
00A0 01DB 01DB 01DB 00E7 00E7 00E7  051E 12796          .WORD   ^X00E7,^X00E7,^X00E7,^X01DB,^X01DB,^X01DB,^X00A0 ; 69 - 6F
036E 022C 00F8 03F8 0315 036C 0315  052C 12797          .WORD   ^X0315,^X036C,^X0315,^X03F8,^X00F8,^X022C,^X036E ; 70 - 76
01E5 03F8 0064 0018 0092 0092 03F8  053A 12798          .WORD   ^X03F8,^X0092,^X0092,^X0018,^X0064,^X03F8,^X01E5 ; 77 - 7D
02A7 025B 025B 023D 023D 03F8 01EE  0548 12799          .WORD   ^X01EE,^X03F8,^X023D,^X023D,^X025B,^X025B,^X02A7 ; 7E - 84
0291 0291 028B 028B 008C 0309 001C  0556 12800          .WORD   ^X001C,^X0309,^X008C,^X028B,^X028B,^X0291,^X0291 ; 85 - 8B
01AF 0052 00EE 03A3 01CF 0279 0279  0564 12801          .WORD   ^X0279,^X0279,^X01CF,^X03A3,^X00EE,^X0052,^X01AF ; 8C - 92
0051 0051 03F8 03F8 03F8 03F8 000A  0572 12802          .WORD   ^X000A,^X03F8,^X03F8,^X03F8,^X03F8,^X0051,^X0051 ; 93 - 99
023D 03F8 00EE 039F 0211 00EE 00EE  0580 12803          .WORD   ^X00EE,^X00EE,^X0211,^X039F,^X00EE,^X03F8,^X023D ; 9A - A0
```

```
 008C 0309 001C 02A7 025B 025B 023D  058E 12804          .WORD   ^X023D,^X025B,^X025B,^X02A7,^X001C,^X0309,^X008C ; A1 - A7
 01CF 0279 0279 0291 0291 028B 028B  059C 12805          .WORD   ^X028B,^X028B,^X0291,^X0291,^X0279,^X0279,^X01CF ; A8 - AE
 03F8 03F8 000A 01AF 0052 00EE 03A3  05AA 12806          .WORD   ^X03A3,^X00EE,^X0052,^X01AF,^X000A,^X03F8,^X03F8 ; AF - B5
 03F8 03F8 03F8 03F8 03F8 03F8 03F8  05B8 12807          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; B6 - BC
 025B 025B 023D 023D 03F8 03F8 03F8  05C6 12808          .WORD   ^X03F8,^X03F8,^X03F8,^X023D,^X023D,^X025B,^X025B ; BD - C3
 0291 028B 028B 008C 0309 001C 02A7  05D4 12809          .WORD   ^X02A7,^X001C,^X0309,^X008C,^X028B,^X028B,^X0291 ; C4 - CA
 0052 00EE 03A3 01CF 0279 0279 0291  05E2 12810          .WORD   ^X0291,^X0279,^X0279,^X01CF,^X03A3,^X00EE,^X0052 ; CB - D1
 0231 03F8 03F8 03F8 03F8 000A 01AF  05F0 12811          .WORD   ^X01AF,^X000A,^X03F8,^X03F8,^X03F8,^X03F8,^X0231 ; D2 - D8
 03F8 00EE 03F8 03F8 00EE 0390 0271  05FE 12812          .WORD   ^X0271,^X0390,^X00EE,^X03F8,^X03F8,^X00EE,^X03F8 ; D9 - DF
 0287 0287 0287 0287 0287 0287 0287  060C 12813          .WORD   ^X0287,^X0287,^X0287,^X0287,^X0287,^X0287,^X0287 ; E0 - E6
 024F 026F 01EF 00EF 03F8 03F8 0287  061A 12814          .WORD   ^X0287,^X03F8,^X03F8,^X00EF,^X01EF,^X026F,^X024F ; E7 - ED
 03F8 03A1 03A1 039F 0280 020F 022F  0628 12815          .WORD   ^X022F,^X020F,^X0280,^X039F,^X03A1,^X03A1,^X03F8 ; EE - F4
 03A5 03A5 031E 02B4 00AE 00AE 03F8  0636 12816          .WORD   ^X03F8,^X00AE,^X00AE,^X02B4,^X031E,^X03A5,^X03A5 ; F5 - FB
           03F8 03F8 0776 0776       0644 12817          .WORD   ^X0776,^X0776,^X03F8,^X03F8                      ; FC - FF
                                     064C 12818
                                     064C 12819 BEGIN_CPU_DATA
                                     0002 12820
                                     0002 12821 DCS_DATA
                                     0001 12822
U 00. 7700,C800,0364,0300,0470,1801  0001 12823 ;% 00   NOP                                  ; get cpu up to speed
U 01. 5700,C840,0364,0308,0470,1802  000C 12827 ;% 01   R[TEMP2]_RDM                         ; load opcode in R[TEMP2]
U 02. 7700,F800,56D7,57FF,8470,1803  0017 12831 ;% 02   LONLIT_[52515000]                   ; load opspecs in lonlit reg.
U 03. 7700,8800,5BE5,03D4,0470,1804  0022 12835 ;% 03   Q_R[LONLIT]                        ; load Q with opspecs
U 04. 7700,0842,00A4,0308,0470,1805  002D 12839 ;% 04   R[TEMP2]_M[TEMP2].OR.Q              ; OR opcode and opspecs
U 05. 7700,4868,5BE4,0308,0470,1806  0038 12843 ;% 05   M[TEMP8]_R[TEMP2]                   ; save in M[TEMP8] for error
U 06. 7700,C800,5BE4,03D8,4A70,1807  0043 12847 ;% 06   VA_R[ZERO]                        ; zero the VA register
U 07. 7701,C800,5BE4,0208,5590,1808  004E 12851 ;% 07   WRITE.LONG R[TEMP2]                ; write data for XB0
U 08. 7700,0800,5BE4,0308,5470,1809  0059 12855 ;% 08   WDR_UNROT(R[TEMP2])               ; validate cache location 0
U 09. 7700,C800,0364,0300,4470,180A  0064 12859 ;% 09   VA_VA+4                           ; update the VA by 4
U 0A. 7700,3800,54D5,55D6,0470,180B  006F 12863 ;% 0A   LONLIT_[56555453]                 ; load opspecs in lonlit reg.
U 0B. 7700,4869,5BE4,03D4,0470,180C  007A 12867 ;% 0B   M[TEMP9]_R[LONLIT]                ; save in M[TEMP9] for error
U 0C. 7701,8800,5BE4,02D4,5590,180D  0085 12871 ;% 0C   WRITE.LONG R[LONLIT]              ; write data for XB1
U 0D. 7700,C800,5BE4,03D4,5470,180E  0090 12875 ;% 0D   WDR_UNROT(R[LONLIT])             ; validate cache location 4
U 0E. 7700,C800,5BE4,03D8,4870,180F  009B 12879 ;% 0E   PC_R[ZERO]                       ; zero PC to force prefetch
U 0F. 7700,C800,0364,1700,0470,1810  00A6 12883 ;% 0F   BUT/IRD1TST                      ; perform ird1
U 10. 7300,4800,0364,0300,0470,1811  00B1 12887 ;% 10   NOP,STOP.CLOCK                    ; halt
                                     00BC 12891
U 11. 7700,4800,0364,0300,0470,1812  00BC 12892 ;% 11   NOP                              ; get cpu up to speed
U 12. 7700,4800,0364,1B00,0470,1800  00C7 12896 ;% 12   BUT/LOD.INC.BRA,NEXT/1800        ; load OSR & increment IRD cntr
U 13. 7300,4800,0364,0300,0470,1814  00D2 12900 ;% 13   STOP.CLOCK                       ; halt
                                     00DD 12904
U 14. 7700,C800,0364,0300,0470,1815  00DD 12905 ;% 14   NOP                              ; get cpu up to speed
U 15. 5700,C800,0364,0300,2A70,1816  00E8 12909 ;% 15   WCTRL/FPA.ENABLE_WB5,WB_RDM       ; write FPA enable bit
U 16. 7700,4800,0364,0300,0470,1817  00F3 12913 ;% 16   NOP                              ; ...
U 17. 7700,4800,0364,C300,0470,1800  00FE 12917 ;% 17   BUT/NO.FPA,NEXT/1800            ; check FPA status
U 18. 7300,C800,0364,0300,0470,1819  0109 12921 ;% 18   STOP.CLOCK                       ; halt
                                     0114 12925
U 19. 7700,C800,0364,0300,0470,181A  0114 12926 ;% 19   NOP                              ; get the cpu up to speed
U 1A. 7700,F800,7F9F,FFFF,8470,181B  011F 12930 ;% 1A   LONLIT_[00C00000]                 ; address of s&c bit in crar
U 1B. 7700,4800,5BE4,03D4,2470,181C  012A 12934 ;% 1B   WCTRL/LOADCRAR,WB_R[LONLIT]      ; address the s&c bit
U 1C. 7700,B800,7FDF,FFFF,8470,181D  0135 12938 ;% 1C   LONLIT_[00400000]                 ; data for ctrl-p int. disable
U 1D. 7700,4800,5BE4,03D4,2C70,181E  0140 12942 ;% 1D   WCTRL/CONWRITE,WB_R[LONLIT]      ; disable ctrl-p interrupts
U 1E. 7300,C800,0364,0300,0470,181F  014B 12946 ;% 1E   STOP.CLOCK                       ; halt
                                     0156 12950
U 1F. 7700,C800,0364,0300,0470,1820  0156 12951 ;% 1F   NOP                              ; get the cpu up to speed
```

```
U 20. 7700,F800,7F9F,FFFF,8470,1821  0161 12955 ;% 20   LONLIT_[00C00000]                    ; address of s&c bit in crar
U 21. 7700,C800,5BE4,03D4,2470,1822  016C 12959 ;% 21   WCTRL/LOADCRAR,WB_R[LONLIT]         ; address the s&c bit
U 22. 7700,7800,7FB7,FFFF,8470,1823  0177 12963 ;% 22   LONLIT_[00900000]                    ; data for ctrl-p int. enable
U 23. 7700,4800,5BE4,03D4,2C70,1824  0182 12967 ;% 23   WCTRL/CONWRITE,WB_R[LONLIT]         ; enable ctrl-p interrupts
U 24. 7300,C800,0364,0300,0470,1825  018D 12971 ;% 24   STOP.CLOCK                         ; halt
                                     0198 12975
                                     0198 12976 ; Set-up RTEMP0 and RTEMP1 for error callout
                                     0198 12977
U 25. 7700,C800,0364,0300,0470,1826  0198 12978 ;% 25   NOP                                ; get cpu up to speed
U 26. 7700,0848,5924,0300,0470,1827  01A3 12982 ;% 26   R[TEMP0]_M[TEMP8]                  ; move XB0 data to R[TEMP0]
U 27. 7700,0849,5924,0304,0470,1828  01AE 12986 ;% 27   R[TEMP1]_M[TEMP9]                  ; move XB1 data to R[TEMP1]
U 28. 7300,C800,0364,0300,0470,1829  01B9 12990 ;% 28   STOP.CLOCK                         ; halt
                                     01C4 12994
                                     01C4 12995 RTEMP_DATA
                                     0001 12996
                          00000000   0001 12997         .LONG   0                  ; rtemp0 = contents of xb0
                          00000000   0005 12998         .LONG   0                  ; rtemp1 = contents of xb1
                          00000000   0009 12999         .LONG   0                  ; rtemp2 = instruction
                                     000D 13000
                                     000D 13001 END_CPU_DATA
                                     064C 13002
                                     064C 13003 END_TEST_DATA
                                     064C
                                     064C        ;-------------------------------------------------------------------------
                                     064C 13004
                                     064C 13005 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
                        0032        .SBTTL  "TEST   0CD     Second IRDX Instruction Address Decode Test 4
                        0032 13007 ;*******************************************************************************
                        0032 13008 ;++
                        0032 13009 ;
                        0032 13010 ; FUNCTIONAL DESCRIPTION
                        0032 13011 ;
                        0032 13012 ;       This test will examine the signal ROM OS INH H from the Native Mode
                        0032 13013 ;       IRDX roms when the IRD Counter is equal to one and the operand
                        0032 13014 ;       specifier is for a register reference.
                        0032 13015 ;
                        0032 13016 ; TEST ALGORITHM
                        0032 13017 ;
                        0032 13018 ;       This test consists of three subtests.  The first subtest will examine
                        0032 13019 ;       the IRDX roms when the Floating Point Accelerator (FPA) is not present
                        0032 13020 ;       in the system.  The second subtest will examine the IRDX roms when the
                        0032 13021 ;       FPA is present but disabled.  The third subtest will examine the IRDX
                        0032 13022 ;       roms when the FPA is present and enabled.  The algorithm is as follows:
                        0032 13023 ;
                        0032 13024 ;        1.  Initialize the processor.
                        0032 13025 ;        2.  Disable CTRL/P interrupts.
                        0032 13026 ;        3.  Check to see if FPA present
                        0032 13027 ;               If FPA present
                        0032 13028 ;                   Then skip to Subtest 2
                        0032 13029 ;               Else fall through to subtest 1
                        0032 13030 ;
                        0032 13031 ;       Subtest 1
                        0032 13032 ;
                        0032 13033 ;        4.  Set-up flag to expect a micro trap.
                        0032 13034 ;        5.  Initialize loop I for 256 opcodes (00-FF).
                        0032 13035 ;        6.  Store the index I in a location that will be used to load into the
                        0032 13036 ;            WDREG.  The value is used as the opcode.
                        0032 13037 ;        7.  Loop on error from here.
                        0032 13038 ;        8.  Set-up DCS location 12 with a BUT/LOD.INC.BRA?,NEXT/1800.  This
                        0032 13039 ;            is used to get the IRD counter to 1 without sourcing an address
                        0032 13040 ;            from the IRDX roms (ie., by doing an IRDX).
                        0032 13041 ;        9.  Execute the following test microcode (performs the IRD1):
                        0032 13042 ;               a.  NOP to get the RDM up to speed.
                        0032 13043 ;               b.  Read the WDREG into RTEMP2 (opcode).
                        0032 13044 ;               c.  Load the operands into the LONLIT register.
                        0032 13045 ;               d.  Store the operands into the Q register to prepare for ORing
                        0032 13046 ;                   the opcode with the operands.
                        0032 13047 ;               e.  OR the opcode and operands together (XB0 data).
                        0032 13048 ;               f.  Store XB0 data in M[TEMP8] in case needed for error callout.
                        0032 13049 ;               g.  Zero the VA.
                        0032 13050 ;               h.  Validate cache location 0 by first writing the data in
                        0032 13051 ;                   RTEMP2 then reading the data back.
                        0032 13052 ;               i.  Update the VA by 4.
                        0032 13053 ;               j.  Load the second operand data into the LONLIT register
                        0032 13054 ;                   (XB1 data).
                        0032 13055 ;               k.  Store XB1 data in M[TEMP9] in case needed for error callout.
                        0032 13056 ;               l.  Validate cache location 4 by first writing the data in
                        0032 13057 ;                   the LONLIT register then reading the data back.
                        0032 13058 ;               m.  Load the PC with zero to cause a prefetch to take place.
                        0032 13059 ;               n.  Do an IRD1 to init the IRD counter and prepare for instruc-
                        0032 13060 ;                   tion/operand specifier evaluation.
```

```
0032 13061 ;                    o.  Halt.
0032 13062 ;            10.  Perform the following test microcode (performs the LOD.INC.BRA?)
0032 13063 ;                    a.  NOP to get the RDM up to speed.
0032 13064 ;                    b.  Do an BUT/LOD.INC.BRA?,NEXT/1800
0032 13065 ;                    c.  Halt.
0032 13066 ;            11.  Load DCS location 12 with a BUT/IRDX micro order.
0032 13067 ;            12.  Perform the following test microcode (performs the IRDX with the
0032 13068 ;                 IRD counter = 1):
0032 13069 ;                    a.  NOP to get the RDM up to speed.
0032 13070 ;                    b.  Do an IRDX and strobe the VBUS.
0032 13071 ;                    c.  Halt.
0032 13072 ;            13.  Compare the VBUS data with the expected value and bit position
0032 13073 ;                 from the VBUS Table.
0032 13074 ;            14.  If the two do not equal, execute the following microcode and
0032 13075 ;                 signal an error (this is done to set-up for printing RTEMP0 & 1
0032 13076 ;                 on the error callout):
0032 13077 ;                    a.  NOP to get the RDM up to speed.
0032 13078 ;                    b.  Load RTEMP0 with the contents of M[TEMP8]
0032 13079 ;                    c.  Load RTEMP1 with the contents of M[TEMP9]
0032 13080 ;                    d.  Halt
0032 13081 ;            15.  Continue with loop I for the 256 VAX opcodes.
0032 13082 ;            16.  Re-enable CTRL/P interrupts then skip to next tests.
0032 13083 ;
0032 13084 ;            Subtest 2
0032 13085 ;
0032 13086 ;            17.  Load the FPA disable bit (bit <5> equal to 0).
0032 13087 ;            18.  Loop on error from here.
0032 13088 ;            19.  Set-up to expect micro trap.
0032 13089 ;            20.  Execute the test microcode.  This is done only to get the FPA to
0032 13090 ;                 a state where it can be disabled or enabled.  If this had not been
0032 13091 ;                 done before then the FPA would be setting in the initialize state
0032 13092 ;                 which forces FPA enable.
0032 13093 ;            21.  Execute the following microcode to disable the FPA:
0032 13094 ;                    a.  NOP to get the RDM up to speed.
0032 13095 ;                    b.  Read the RDM WDREG onto the W Bus and then write the W Bus
0032 13096 ;                        to the FPA.
0032 13097 ;                    c.  Perform a NO.FPA test to insure that the FPA got disabled.
0032 13098 ;                    d.  Halt.
0032 13099 ;            22.  Check the Trap register to insure that the FPA got disabled.
0032 13100 ;            23.  If the FPA is not disabled, then signal an error.
0032 13101 ;            24.  Set-up flag to expect a micro trap.
0032 13102 ;            25.  Initialize loop I for 256 opcodes (00-FF).
0032 13103 ;            26.  Store the index I in a location that will be used to load into the
0032 13104 ;                 WDREG.  The value is used as the opcode.
0032 13105 ;            27.  Loop on error from here.
0032 13106 ;            28.  Set-up DCS location 12 with a BUT/LOD.INC.BRA?,NEXT/1800.  This
0032 13107 ;                 is used to get the IRD counter to 1 without sourcing an address
0032 13108 ;                 from the IRDX roms (ie., by doing an IRDX).
0032 13109 ;            29.  Execute the following test microcode:
0032 13110 ;                    a.  NOP to get the RDM up to speed.
0032 13111 ;                    b.  Read the WDREG into RTEMP2 (opcode).
0032 13112 ;                    c.  Load the operands into the LONLIT register.
0032 13113 ;                    d.  Store the operands into the Q register to prepare for ORing
0032 13114 ;                        the opcode with the operands.
0032 13115 ;                    e.  OR the opcode and operands together (XB0 data).
```

```
0032 13116 ;                f.  Store XB0 data in M[TEMP8] in case needed for error callout.
0032 13117 ;                g.  Zero the VA.
0032 13118 ;                h.  Validate cache location 0 by first writing the data in
0032 13119 ;                    RTEMP2 then reading the data back.
0032 13120 ;                i.  Update the VA by 4.
0032 13121 ;                j.  Load the second operand data into the LONLIT register
0032 13122 ;                    (XB1 data).
0032 13123 ;                k.  Store XB1 data in M[TEMP9] in case needed for error callout.
0032 13124 ;                l.  Validate cache location 4 by first writing the data in
0032 13125 ;                    the LONLIT register then reading the data back.
0032 13126 ;                m.  Load the PC with zero to cause a prefetch to take place.
0032 13127 ;                n.  Do an IRD1 to init the IRD counter and prepare for instruc-
0032 13128 ;                    tion/operand specifier evaluation.
0032 13129 ;                o.  Halt.
0032 13130 ;        30.  Perform the following test microcode.
0032 13131 ;                a.  NOP to get the RDM up to speed.
0032 13132 ;                b.  Do an BUT/LOD.INC.BRA?,NEXT/1800
0032 13133 ;                c.  Halt.
0032 13134 ;        31.  Load DCS location 12 with a BUT/IRDX micro order.
0032 13135 ;        32.  Perform the following test microcode (performs the IRDX with the
0032 13136 ;             IRD counter = 1):
0032 13137 ;                a.  NOP to get the RDM up to speed.
0032 13138 ;                b.  Do an IRDX and strobe the VBUS.
0032 13139 ;                c.  Halt.
0032 13140 ;        33.  Compare the VBUS data with the expected value and bit position
0032 13141 ;             from the VBUS Table.
0032 13142 ;        34.  If the two do not equal, execute the following microcode and
0032 13143 ;             signal an error (this is done to set-up for printing RTEMP0 & 1
0032 13144 ;             on the error callout):
0032 13145 ;                a.  NOP to get the RDM up to speed.
0032 13146 ;                b.  Load RTEMP0 with the contents of M[TEMP8]
0032 13147 ;                c.  Load RTEMP1 with the contents of M[TEMP9]
0032 13148 ;                d.  Halt
0032 13149 ;        35.  Continue with loop I for the 256 VAX opcodes.
0032 13150 ;        36.  Go to Subtest 3.
0032 13151 ;
0032 13152 ;        Subtest 3
0032 13153 ;
0032 13154 ;        37.  Load the FPA enable bit (bit <5> equal to 1).
0032 13155 ;        38.  Loop on error from here.
0032 13156 ;        39.  Set-up to expect micro trap.
0032 13157 ;        40.  Execute the following microcode to enable the FPA:
0032 13158 ;                a.  NOP to get the RDM up to speed.
0032 13159 ;                b.  Read the RDM WDREG onto the W Bus and then write the W Bus
0032 13160 ;                    to the FPA.
0032 13161 ;                c.  Perform a NO.FPA test to insure that the FPA got enabled.
0032 13162 ;                d.  Halt.
0032 13163 ;        41.  Check the Trap register to insure that the FPA got enabled.
0032 13164 ;        42.  If the FPA is not enabled, then signal an error.
0032 13165 ;        43.  Set-up flag to expect a micro trap.
0032 13166 ;        44.  Initialize loop I for 256 opcodes (00-FF).
0032 13167 ;        45.  Store the index I in a location that will be used to load into the
0032 13168 ;             WDREG.  The value is used as the opcode.
0032 13169 ;        46.  Loop on error from here.
0032 13170 ;        47.  Set then Clear the DCLO bit in the RDCTRL register to perform an
```

```
0032 13171 ;            initialize.  This is done because when the test microcode
0032 13172 ;            executes, if the opcode was an FP opcode then there is a
0032 13173 ;            possibility that the processor could be setting there in an
0032 13174 ;            FPAWAIT state (M clock and D clock stalled).  If this were to
0032 13175 ;            happen then the cpu clocks would no longer run and the remaining
0032 13176 ;            test opcodes would fail.
0032 13177 ;       48.  Set-up DCS location 12 with a BUT/LOD.INC.BRA?,NEXT/1800.  This
0032 13178 ;            is used to get the IRD counter to 1 without sourcing an address
0032 13179 ;            from the IRDX roms (ie., by doing an IRDX).
0032 13180 ;       49.  Execute the following test microcode:
0032 13181 ;               a.  NOP to get the RDM up to speed.
0032 13182 ;               b.  Read the WDREG into RTEMP2 (opcode).
0032 13183 ;               c.  Load the operands into the LONLIT register.
0032 13184 ;               d.  Store the operands into the Q register to prepare for ORing
0032 13185 ;                   the opcode with the operands.
0032 13186 ;               e.  OR the opcode and operands together (XB0 data).
0032 13187 ;               f.  Store XB0 data in M[TEMP8] in case needed for error callout.
0032 13188 ;               g.  Zero the VA.
0032 13189 ;               h.  Validate cache location 0 by first writing the data in
0032 13190 ;                   RTEMP2 then reading the data back.
0032 13191 ;               i.  Update the VA by 4.
0032 13192 ;               j.  Load the second operand data into the LONLIT register
0032 13193 ;                   (XB1 data).
0032 13194 ;               k.  Store XB1 data in M[TEMP8] in case needed for error callout.
0032 13195 ;               l.  Validate cache location 4 by first writing the data in
0032 13196 ;                   the LONLIT register then reading the data back.
0032 13197 ;               m.  Load the PC with zero to cause a prefetch to take place.
0032 13198 ;               n.  Do an IRD1 to init the IRD counter and prepare for instruc-
0032 13199 ;                   tion/operand specifier evaluation.
0032 13200 ;               o.  Halt.
0032 13201 ;       50.  Perform the following test microcode.
0032 13202 ;               a.  NOP to get the RDM up to speed.
0032 13203 ;               b.  Do an BUT/LOD.INC.BRA?,NEXT/1800
0032 13204 ;               c.  Halt.
0032 13205 ;       51.  Load DCS location 12 with a BUT/IRDX micro order.
0032 13206 ;       52.  Perform the following test microcode (performs the IRDX with the
0032 13207 ;            IRD counter = 1):
0032 13208 ;               a.  NOP to get the RDM up to speed.
0032 13209 ;               b.  Do an IRDX and strobe the VBUS.
0032 13210 ;               c.  Halt.
0032 13211 ;       53.  Compare the VBUS data with the expected value and bit position
0032 13212 ;            from the VBUS Table.
0032 13213 ;       54.  If the two do not equal, execute the following microcode and
0032 13214 ;            signal an error (this is done to set-up for printing RTEMP0 & 1
0032 13215 ;            on the error callout):
0032 13216 ;               a.  NOP to get the RDM up to speed.
0032 13217 ;               b.  Load RTEMP0 with the contents of M[TEMP8]
0032 13218 ;               c.  Load RTEMP1 with the contents of M[TEMP9]
0032 13219 ;               d.  Halt
0032 13220 ;       55.  Continue with loop I for the 256 VAX opcodes.
0032 13221 ;       56.  Re-enable CTRL/P interrupts.
0032 13222 ;       57.  Skip to next tests.
0032 13223 ;
0032 13224 ; TEST PATTERNS
0032 13225 ;
```

```
                        0032 13226 ;       For this test the opcodes are generated by a simple count pattern.  The
                        0032 13227 ;       opcodes are based on the 256 base machine instructions as specified in
                        0032 13228 ;       the VAX architecture handbook.  The generated opcodes are written to
                        0032 13229 ;       the CPU and combined (ORed) with three bytes of operand data to form
                        0032 13230 ;       a machine instruction.  The operands all specify Register Deferred mode
                        0032 13231 ;       using registers 0-6 and are loaded from the LONLIT register.
                        0032 13232 ;
                        0032 13233 ;       Also required are two tables of expected VBUS values for the second
                        0032 13234 ;       IRDX cycle.  One table contains the VBUS values for the second IRDX
                        0032 13235 ;       when the FPA is disabled, the other for when it is enabled.
                        0032 13236 ;
                        0032 13237 ; LOGIC DESCRIPTION
                        0032 13238 ;
                        0032 13239 ;       This test will check the validity of the IRDX roms on the DPM module.
                        0032 13240 ;       The IRDX roms are addressed by bits <7:0> of the Instruction Register.
                        0032 13241 ;       Along with being addressed by the IR, there are three other signals
                        0032 13242 ;       that affect the address.  They are IRD CTR 0 H, FPA ENABLED L, and
                        0032 13243 ;       REG MODE H.  This test will examine every 8th(hex) address starting at
                        0032 13244 ;       address 6 with FPA disabled and starting at address 4 with FPA enabled.
                        0032 13245 ;
                        0032 13246 ; ASSUMPTIONS
                        0032 13247 ;
                        0032 13248 ;       This test assumes the following logic has already been tested; the
                        0032 13249 ;       CPU's data path, the cache, the XB, and the IR and OSR.
                        0032 13250 ;
                        0032 13251 ; ERROR DESCRIPTION
                        0032 13252 ;
                        0032 13253 ;       Subtest 1
                        0032 13254 ;
                        0032 13255 ;               Expected CS Address
                        0032 13256 ;               Received CS Address
                        0032 13257 ;
                        0032 13258 ;       Subtest 2
                        0032 13259 ;
                        0032 13260 ;               First IFERROR
                        0032 13261 ;
                        0032 13262 ;                       Expected CS Address
                        0032 13263 ;                       Received CS Address
                        0032 13264 ;
                        0032 13265 ;               Second IFERROR
                        0032 13266 ;
                        0032 13267 ;                       Expected VBUS ( 00000ynn )      y  = VBUS bit value
                        0032 13268 ;                       Received VBUS ( 00000ynn )      nn = VBUS bit number
                        0032 13269 ;                       Loop I count
                        0032 13270 ;                       Contents of XB0
                        0032 13271 ;                       Contents of XB1
                        0032 13272 ;
                        0032 13273 ;       Subtest 3
                        0032 13274 ;
                        0032 13275 ;               First IFERROR
                        0032 13276 ;
                        0032 13277 ;                       Expected CS Address
                        0032 13278 ;                       Received CS Address
                        0032 13279 ;
                        0032 13280 ;               Second IFERROR
```

```
                        0032 13281 ;
                        0032 13282 ;                    Expected VBUS ( 00000ynn )      y  = VBUS bit value
                        0032 13283 ;                    Received VBUS ( 00000ynn )      nn = VBUS bit number
                        0032 13284 ;                    Loop I count
                        0032 13285 ;                    Contents of XB0
                        0032 13286 ;                    Contents of XB1
                        0032 13287 ;--
                        0032 13288 ;*****************************************************************************
                        0032 13289
                        0032 13290          INITIALIZE                              ; initialize the comet cpu
                        0034 13291          FETCH           <^X19>                  ; disable ctrl-p interrupts
                        0039 13292          BRSTCLK         <^X1E>                  ; ...
                        003D 13293          FETCH           <^X16>                  ; execute a but/no.fpa to check
                        0042 13294          BRSTCLK         <^X18>                  ; for fpa present
                        0046 13295          CMPREGMSK       CSTRP,1$,,2$,,W         ; is the fpa present?
                        0052 13296          SKIP            100$,ONERROR            ; if so then skip to subtest 2
                        0059 13297
                        0059 13298          SUBTEST         1
                        005C
                        005C            ;////////////////////////////////////////////////////////////////////////////
                        005C 13299 ;+
                        005C 13300 ;
                        005C 13301 ; FPA is not present.
                        005C 13302 ;
                        005C 13303 ;-
                        005C 13304          EXPUTRAP                                ; else expect non-dcs addresses
                        005E 13305          LOOP            I,1,256                 ; loop for 256 opcodes
                        0065 13306          SAVEINDEX       I,9$                    ; save the loop count
                        006A 13307          LOADREG         WDREG,9$,,L             ; load wdreg with loop count
                        0072 13308          ERRLOOP                                 ; loop on error from here
                        0074 13309          LOADDCS         20$,,<^X12>,1           ; load BUT/LOD.INC.BRA?
                        007B 13310          FETCH           0                       ; execute ird1tst
                        0080 13311          BRSTCLK         <^X10>,INH              ; ...
                        0084 13312          FETCH           <^X11>                  ; execute BUT/LOD.INC.BRA?
                        0089 13313          BRSTCLK         <^X13>,INH              ; ...
                        008D 13314          LOADDCS         21$,,<^X12>,1           ; load BUT/IRDX
                        0094 13315          FETCH           <^X11>                  ; execute the second irdx
                        0099 13316          BRSTCLK         <^X12>,INH              ; ...
                        009D 13317          CMPVBUS         30$,I                   ; check VBUS bit for an error
                        00A2 13318          SKIP            50$,NOERROR             ; continue testing if no error
                        00A9 13319          FETCH           <^X25>                  ; start dcs program
                        00AE 13320          BRSTCLK         <^X28>                  ; end dcs program
                        00B2 13321          IFERROR         2,<<IRD>>,<<MIC>>       ; signal an error
                        00B9 13322 50$:     ENDLOOP         I                       ; continue loop
                        00BC 13323          FETCH           <^X1F>                  ; re-enable ctrl-p interrupts
                        00C1 13324          BRSTCLK         <^X24>                  ; ...
                        00C5 13325          SKIP                                    ; skip on to the next test
                        00CC 13326
                        00CC 13327 100$:    SUBTEST         2
                        00CF
                        00CF            ;////////////////////////////////////////////////////////////////////////////
                        00CF 13328 ;+
                        00CF 13329 ;
                        00CF 13330 ; FPA present but not enabled
                        00CF 13331 ;
```

```
00CF 13332 ;-
00CF 13333          LOADREG         WDREG,7$,,L             ; load fpa disable bit
00D7 13334          ERRLOOP                                 ; loop on error here
00D9 13335          EXPUTRAP                                ; expect micro trap
00DB 13336          FETCH           0                       ; start dcs program
00E0 13337          BRSTCLK         <^X10>,INH              ; end dcs program
00E4 13338          FETCH           <^X14>                  ; execute the disable
00E9 13339          BRSTCLK         <^X18>,INH              ; and read back fpa status
00ED 13340          CMPREGMSK       CSTRP,3$,,5$,,W         ; is the fpa disabled?
00F9 13341          IFERROR         ,,<FPA>                 ; if not, signal an error
00FF 13342          EXPUTRAP                                ; else expect non-dcs addresses
0101 13343          LOOP            I,1,256                 ; loop for 256 opcodes
0108 13344          SAVEINDEX       I,9$                    ; save the loop count
010D 13345          LOADREG         WDREG,9$,,L             ; load wdreg with loop count
0115 13346          ERRLOOP                                 ; loop on error from here
0117 13347          LOADDCS         20$,,<^X12>,1           ; load BUT/LOD.INC.BRA?
011E 13348          FETCH           0                       ; execute ird1tst
0123 13349          BRSTCLK         <^X10>,INH              ; ...
0127 13350          FETCH           <^X11>                  ; execute BUT/LOD.INC.BRA?
012C 13351          BRSTCLK         <^X13>,INH              ; ...
0130 13352          LOADDCS         21$,,<^X12>,1           ; load BUT/IRDX
0137 13353          FETCH           <^X11>                  ; execute the second irdx
013C 13354          BRSTCLK         <^X13>,INH              ; ...
0140 13355          CMPVBUS         30$,I                   ; check VBUS bit for an error
0145 13356          SKIP            51$,NOERROR             ; continue testing if no error
014C 13357          FETCH           <^X25>                  ; start dcs program
0151 13358          BRSTCLK         <^X28>                  ; end dcs program
0155 13359          IFERROR         2,<<IRD>>,<<MIC>>       ; signal an error
015C 13360 51$:     ENDLOOP         I                       ; continue loop
015F 13361
015F 13362          SUBTEST         3
0162
0162          ;////////////////////////////////////////////////////////////////////////////////
0162 13363 ;+
0162 13364 ;
0162 13365 ; FPA present and enabled
0162 13366 ;
0162 13367 ;-
0162 13368          LOADREG         WDREG,8$,,L             ; load fpa enable bit
016A 13369          ERRLOOP                                 ; loop on error from here
016C 13370          FETCH           <^X14>                  ; enable the fpa
0171 13371          BRSTCLK         <^X18>,INH              ; and read back fpa enable
0175 13372          CMPREGMSK       CSTRP,4$,,5$,,W         ; is the fpa enabled?
0181 13373          IFERROR         ,,<FPA>                 ; if not, signal an error
0187 13374          EXPUTRAP                                ; else expect non-dcs addresses
0189 13375          LOOP            I,1,256                 ; loop for 256 opcodes
0190 13376          SAVEINDEX       I,9$                    ; save the loop count
0195 13377          LOADREG         WDREG,9$,,L             ; load wdreg with loop count
019D 13378          ERRLOOP                                 ; loop on error from here
019F 13379          LOADDCS         20$,,<^X12>,1           ; load BUT/LOD.INC.BRA?
01A6 13380          LOADREG         RDCTRL,10$,,B           ; assert dclo
01AE 13381          LOADREG         RDCTRL,11$,,B           ; de-assert dclo
01B6 13382          FETCH           0                       ; execute ird1tst
01BB 13383          BRSTCLK         <^X10>,INH              ; ...
01BF 13384          FETCH           <^X11>                  ; execute BUT/LOD.INC.BRA?
```

```
                                01C4          BRSTCLK        <^X13>,INH                ; ...
                                01C8          LOADDCS        21$,,<^X12>,1             ; load BUT/IRDX
                                01CF          FETCH          <^X11>                    ; execute the second irdx
                                01D4          BRSTCLK        <^X13>,INH                ; ...
                                01D8          CMPVBUS        31$,I                     ; check VBUS bit for an error
                                01DD          SKIP           52$,NOERROR               ; continue testing if no error
                                01E4          FETCH          <^X25>                    ; start dcs program
                                01E9          BRSTCLK        <^X28>                    ; end dcs program
                                01ED          IFERROR        2,<<IRD>>,<<MIC>>         ; signal an error
                                01F4  52$:    ENDLOOP        I                         ; continue loop
                                01F7          FETCH          <^X1F>                    ; re-enable ctrl-p interrupts
                                01FC          BRSTCLK        <^X24>                    ; ...
                                0200          SKIP                                     ; end of test
                                0207
                                0207  BEGIN_TEST_DATA
                                0207
                                0207      ;-----------------------------------------------------------------------
                                0207
                          1802  0207  1$:     .WORD   ^X1802
                          3FFE  0209  2$:     .WORD   ^X3FFE
                                020B
                          1801  020B  3$:     .WORD   ^X1801
                          1800  020D  4$:     .WORD   ^X1800
                          3FFD  020F  5$:     .WORD   ^X3FFD
                                0211
                          3FFF  0211  6$:     .WORD   ^X3FFF
                                0213
                      00000000  0213  7$:     .LONG   0
                      00000020  0217  8$:     .LONG   ^X00000020
                                021B
                      00000000  021B  9$:     .LONG   0
                            40  021F  10$:    .BYTE   DC_LOW
                            00  0220  11$:    .BYTE   ^X00
                                0221
                                0221
                                0221  ; Loadable microcode
                                0221
                                0221  20$:
U 11, 7700,4800,0364,1B00,0470,1800  0221  ;% 11   BUT/LOD.INC.BRA,NEXT/1800                ; load OSR & increment IRD cntr
                                022C  21$:
U 11, 7600,0800,0364,0700,0470,1812  022C  ;% 11   STROBE.V.BUS,BUT/IRDX                    ; do irdx and strobe VBUS
                                0237
                                0237  ;
                                0237  ; Expected VBUS data for Subtests 1 and 2
                                0237  ;
                                0237  ; The following is a table of all the expected VBUS values for the second IRDX.
                                0237  ; A 1 means the ROM OS INH H signal is asserted and a 0 means it is not
                                0237  ; asserted.
                                0237  ;
                                0237
                            01  0237  30$:    .BYTE   ^X01
                                0238          VBUSDATA       <^X1E>,1                  ;00
                            01  0239          .BYTE   ^X01
                                023A          VBUSDATA       <^X1E>,1                  ;01
                            01  023B          .BYTE   ^X01
```

```
        023C 13444      VBUSDATA        <^X1E>,1                ;02
   01   023D 13445      .BYTE   ^X01
        023E 13446      VBUSDATA        <^X1E>,1                ;03
   01   023F 13447      .BYTE   ^X01
        0240 13448      VBUSDATA        <^X1E>,1                ;04
   01   0241 13449      .BYTE   ^X01
        0242 13450      VBUSDATA        <^X1E>,1                ;05
   01   0243 13451      .BYTE   ^X01
        0244 13452      VBUSDATA        <^X1E>,1                ;06
   01   0245 13453      .BYTE   ^X01
        0246 13454      VBUSDATA        <^X1E>,1                ;07
   01   0247 13455      .BYTE   ^X01
        0248 13456      VBUSDATA        <^X1E>,1                ;08
   01   0249 13457      .BYTE   ^X01
        024A 13458      VBUSDATA        <^X1E>,1                ;09
   01   024B 13459      .BYTE   ^X01
        024C 13460      VBUSDATA        <^X1E>,1                ;0A
   01   024D 13461      .BYTE   ^X01
        024E 13462      VBUSDATA        <^X1E>,1                ;0B
   01   024F 13463      .BYTE   ^X01
        0250 13464      VBUSDATA        <^X1E>,1                ;0C
   01   0251 13465      .BYTE   ^X01
        0252 13466      VBUSDATA        <^X1E>,1                ;0D
   01   0253 13467      .BYTE   ^X01
        0254 13468      VBUSDATA        <^X1E>,1                ;0E
   01   0255 13469      .BYTE   ^X01
        0256 13470      VBUSDATA        <^X1E>,1                ;0F
   01   0257 13471      .BYTE   ^X01
        0258 13472      VBUSDATA        <^X1E>,1                ;10
   01   0259 13473      .BYTE   ^X01
        025A 13474      VBUSDATA        <^X1E>,1                ;11
   01   025B 13475      .BYTE   ^X01
        025C 13476      VBUSDATA        <^X1E>,1                ;12
   01   025D 13477      .BYTE   ^X01
        025E 13478      VBUSDATA        <^X1E>,1                ;13
   01   025F 13479      .BYTE   ^X01
        0260 13480      VBUSDATA        <^X1E>,1                ;14
   01   0261 13481      .BYTE   ^X01
        0262 13482      VBUSDATA        <^X1E>,1                ;15
   01   0263 13483      .BYTE   ^X01
        0264 13484      VBUSDATA        <^X1E>,1                ;16
   01   0265 13485      .BYTE   ^X01
        0266 13486      VBUSDATA        <^X1E>,1                ;17
   01   0267 13487      .BYTE   ^X01
        0268 13488      VBUSDATA        <^X1E>,1                ;18
   01   0269 13489      .BYTE   ^X01
        026A 13490      VBUSDATA        <^X1E>,1                ;19
   01   026B 13491      .BYTE   ^X01
        026C 13492      VBUSDATA        <^X1E>,1                ;1A
   01   026D 13493      .BYTE   ^X01
        026E 13494      VBUSDATA        <^X1E>,1                ;1B
   01   026F 13495      .BYTE   ^X01
        0270 13496      VBUSDATA        <^X1E>,1                ;1C
   01   0271 13497      .BYTE   ^X01
        0272 13498      VBUSDATA        <^X1E>,1                ;1D
```

```
                01  0273 13499          .BYTE   ^X01
                    0274 13500          VBUSDATA        <^X1E>,1                ;1E
                01  0275 13501          .BYTE   ^X01
                    0276 13502          VBUSDATA        <^X1E>,1                ;1F
                01  0277 13503          .BYTE   ^X01
                    0278 13504          VBUSDATA        <^X1E>,1                ;20
                01  0279 13505          .BYTE   ^X01
                    027A 13506          VBUSDATA        <^X1E>,1                ;21
                01  027B 13507          .BYTE   ^X01
                    027C 13508          VBUSDATA        <^X1E>,1                ;22
                01  027D 13509          .BYTE   ^X01
                    027E 13510          VBUSDATA        <^X1E>,1                ;23
                01  027F 13511          .BYTE   ^X01
                    0280 13512          VBUSDATA        <^X1E>,1                ;24
                01  0281 13513          .BYTE   ^X01
                    0282 13514          VBUSDATA        <^X1E>,1                ;25
                01  0283 13515          .BYTE   ^X01
                    0284 13516          VBUSDATA        <^X1E>,1                ;26
                01  0285 13517          .BYTE   ^X01
                    0286 13518          VBUSDATA        <^X1E>,1                ;27
                01  0287 13519          .BYTE   ^X01
                    0288 13520          VBUSDATA        <^X1E>,1                ;28
                01  0289 13521          .BYTE   ^X01
                    028A 13522          VBUSDATA        <^X1E>,1                ;29
                01  028B 13523          .BYTE   ^X01
                    028C 13524          VBUSDATA        <^X1E>,1                ;2A
                01  028D 13525          .BYTE   ^X01
                    028E 13526          VBUSDATA        <^X1E>,1                ;2B
                01  028F 13527          .BYTE   ^X01
                    0290 13528          VBUSDATA        <^X1E>,1                ;2C
                01  0291 13529          .BYTE   ^X01
                    0292 13530          VBUSDATA        <^X1E>,1                ;2D
                01  0293 13531          .BYTE   ^X01
                    0294 13532          VBUSDATA        <^X1E>,1                ;2E
                01  0295 13533          .BYTE   ^X01
                    0296 13534          VBUSDATA        <^X1E>,1                ;2F
                01  0297 13535          .BYTE   ^X01
                    0298 13536          VBUSDATA        <^X1E>,1                ;30
                01  0299 13537          .BYTE   ^X01
                    029A 13538          VBUSDATA        <^X1E>,1                ;31
                01  029B 13539          .BYTE   ^X01
                    029C 13540          VBUSDATA        <^X1E>,1                ;32
                01  029D 13541          .BYTE   ^X01
                    029E 13542          VBUSDATA        <^X1E>,1                ;33
                01  029F 13543          .BYTE   ^X01
                    02A0 13544          VBUSDATA        <^X1E>,1                ;34
                01  02A1 13545          .BYTE   ^X01
                    02A2 13546          VBUSDATA        <^X1E>,1                ;35
                01  02A3 13547          .BYTE   ^X01
                    02A4 13548          VBUSDATA        <^X1E>,1                ;36
                01  02A5 13549          .BYTE   ^X01
                    02A6 13550          VBUSDATA        <^X1E>,1                ;37
                01  02A7 13551          .BYTE   ^X01
                    02A8 13552          VBUSDATA        <^X1E>,1                ;38
                01  02A9 13553          .BYTE   ^X01
```

```
        02AA 13554        VBUSDATA        <^X1E>,1                ;39
   01   02AB 13555        .BYTE   ^X01
        02AC 13556        VBUSDATA        <^X1E>,1                ;3A
   01   02AD 13557        .BYTE   ^X01
        02AE 13558        VBUSDATA        <^X1E>,1                ;3B
   01   02AF 13559        .BYTE   ^X01
        02B0 13560        VBUSDATA        <^X1E>,1                ;3C
   01   02B1 13561        .BYTE   ^X01
        02B2 13562        VBUSDATA        <^X1E>,0                ;3D
   01   02B3 13563        .BYTE   ^X01
        02B4 13564        VBUSDATA        <^X1E>,1                ;3E
   01   02B5 13565        .BYTE   ^X01
        02B6 13566        VBUSDATA        <^X1E>,1                ;3F
   01   02B7 13567        .BYTE   ^X01
        02B8 13568        VBUSDATA        <^X1E>,1                ;40
   01   02B9 13569        .BYTE   ^X01
        02BA 13570        VBUSDATA        <^X1E>,1                ;41
   01   02BB 13571        .BYTE   ^X01
        02BC 13572        VBUSDATA        <^X1E>,1                ;42
   01   02BD 13573        .BYTE   ^X01
        02BE 13574        VBUSDATA        <^X1E>,1                ;43
   01   02BF 13575        .BYTE   ^X01
        02C0 13576        VBUSDATA        <^X1E>,1                ;44
   01   02C1 13577        .BYTE   ^X01
        02C2 13578        VBUSDATA        <^X1E>,1                ;45
   01   02C3 13579        .BYTE   ^X01
        02C4 13580        VBUSDATA        <^X1E>,1                ;46
   01   02C5 13581        .BYTE   ^X01
        02C6 13582        VBUSDATA        <^X1E>,1                ;47
   01   02C7 13583        .BYTE   ^X01
        02C8 13584        VBUSDATA        <^X1E>,1                ;48
   01   02C9 13585        .BYTE   ^X01
        02CA 13586        VBUSDATA        <^X1E>,1                ;49
   01   02CB 13587        .BYTE   ^X01
        02CC 13588        VBUSDATA        <^X1E>,1                ;4A
   01   02CD 13589        .BYTE   ^X01
        02CE 13590        VBUSDATA        <^X1E>,1                ;4B
   01   02CF 13591        .BYTE   ^X01
        02D0 13592        VBUSDATA        <^X1E>,1                ;4C
   01   02D1 13593        .BYTE   ^X01
        02D2 13594        VBUSDATA        <^X1E>,1                ;4D
   01   02D3 13595        .BYTE   ^X01
        02D4 13596        VBUSDATA        <^X1E>,1                ;4E
   01   02D5 13597        .BYTE   ^X01
        02D6 13598        VBUSDATA        <^X1E>,1                ;4F
   01   02D7 13599        .BYTE   ^X01
        02D8 13600        VBUSDATA        <^X1E>,1                ;50
   01   02D9 13601        .BYTE   ^X01
        02DA 13602        VBUSDATA        <^X1E>,1                ;51
   01   02DB 13603        .BYTE   ^X01
        02DC 13604        VBUSDATA        <^X1E>,1                ;52
   01   02DD 13605        .BYTE   ^X01
        02DE 13606        VBUSDATA        <^X1E>,1                ;53
   01   02DF 13607        .BYTE   ^X01
        02E0 13608        VBUSDATA        <^X1E>,1                ;54
```

```
                        01  02E1 13609          .BYTE   ^X01
                            02E2 13610          VBUSDATA        <^X1E>,1                ;55
                        01  02E3 13611          .BYTE   ^X01
                            02E4 13612          VBUSDATA        <^X1E>,1                ;56
                        01  02E5 13613          .BYTE   ^X01
                            02E6 13614          VBUSDATA        <^X1E>,1                ;57
                        01  02E7 13615          .BYTE   ^X01
                            02E8 13616          VBUSDATA        <^X1E>,1                ;58
                        01  02E9 13617          .BYTE   ^X01
                            02EA 13618          VBUSDATA        <^X1E>,1                ;59
                        01  02EB 13619          .BYTE   ^X01
                            02EC 13620          VBUSDATA        <^X1E>,1                ;5A
                        01  02ED 13621          .BYTE   ^X01
                            02EE 13622          VBUSDATA        <^X1E>,1                ;5B
                        01  02EF 13623          .BYTE   ^X01
                            02F0 13624          VBUSDATA        <^X1E>,1                ;5C
                        01  02F1 13625          .BYTE   ^X01
                            02F2 13626          VBUSDATA        <^X1E>,1                ;5D
                        01  02F3 13627          .BYTE   ^X01
                            02F4 13628          VBUSDATA        <^X1E>,1                ;5E
                        01  02F5 13629          .BYTE   ^X01
                            02F6 13630          VBUSDATA        <^X1E>,1                ;5F
                        01  02F7 13631          .BYTE   ^X01
                            02F8 13632          VBUSDATA        <^X1E>,1                ;60
                        01  02F9 13633          .BYTE   ^X01
                            02FA 13634          VBUSDATA        <^X1E>,1                ;61
                        01  02FB 13635          .BYTE   ^X01
                            02FC 13636          VBUSDATA        <^X1E>,1                ;62
                        01  02FD 13637          .BYTE   ^X01
                            02FE 13638          VBUSDATA        <^X1E>,1                ;63
                        01  02FF 13639          .BYTE   ^X01
                            0300 13640          VBUSDATA        <^X1E>,1                ;64
                        01  0301 13641          .BYTE   ^X01
                            0302 13642          VBUSDATA        <^X1E>,1                ;65
                        01  0303 13643          .BYTE   ^X01
                            0304 13644          VBUSDATA        <^X1E>,1                ;66
                        01  0305 13645          .BYTE   ^X01
                            0306 13646          VBUSDATA        <^X1E>,1                ;67
                        01  0307 13647          .BYTE   ^X01
                            0308 13648          VBUSDATA        <^X1E>,1                ;68
                        01  0309 13649          .BYTE   ^X01
                            030A 13650          VBUSDATA        <^X1E>,1                ;69
                        01  030B 13651          .BYTE   ^X01
                            030C 13652          VBUSDATA        <^X1E>,1                ;6A
                        01  030D 13653          .BYTE   ^X01
                            030E 13654          VBUSDATA        <^X1E>,1                ;6B
                        01  030F 13655          .BYTE   ^X01
                            0310 13656          VBUSDATA        <^X1E>,1                ;6C
                        01  0311 13657          .BYTE   ^X01
                            0312 13658          VBUSDATA        <^X1E>,1                ;6D
                        01  0313 13659          .BYTE   ^X01
                            0314 13660          VBUSDATA        <^X1E>,1                ;6E
                        01  0315 13661          .BYTE   ^X01
                            0316 13662          VBUSDATA        <^X1E>,1                ;6F
                        01  0317 13663          .BYTE   ^X01
```

```
                0318 13664          VBUSDATA        <^X1E>,1                ;70
01              0319 13665          .BYTE   ^X01
                031A 13666          VBUSDATA        <^X1E>,1                ;71
01              031B 13667          .BYTE   ^X01
                031C 13668          VBUSDATA        <^X1E>,1                ;72
01              031D 13669          .BYTE   ^X01
                031E 13670          VBUSDATA        <^X1E>,1                ;73
01              031F 13671          .BYTE   ^X01
                0320 13672          VBUSDATA        <^X1E>,1                ;74
01              0321 13673          .BYTE   ^X01
                0322 13674          VBUSDATA        <^X1E>,1                ;75
01              0323 13675          .BYTE   ^X01
                0324 13676          VBUSDATA        <^X1E>,1                ;76
01              0325 13677          .BYTE   ^X01
                0326 13678          VBUSDATA        <^X1E>,1                ;77
01              0327 13679          .BYTE   ^X01
                0328 13680          VBUSDATA        <^X1E>,1                ;78
01              0329 13681          .BYTE   ^X01
                032A 13682          VBUSDATA        <^X1E>,1                ;79
01              032B 13683          .BYTE   ^X01
                032C 13684          VBUSDATA        <^X1E>,1                ;7A
01              032D 13685          .BYTE   ^X01
                032E 13686          VBUSDATA        <^X1E>,1                ;7B
01              032F 13687          .BYTE   ^X01
                0330 13688          VBUSDATA        <^X1E>,1                ;7C
01              0331 13689          .BYTE   ^X01
                0332 13690          VBUSDATA        <^X1E>,1                ;7D
01              0333 13691          .BYTE   ^X01
                0334 13692          VBUSDATA        <^X1E>,1                ;7E
01              0335 13693          .BYTE   ^X01
                0336 13694          VBUSDATA        <^X1E>,1                ;7F
01              0337 13695          .BYTE   ^X01
                0338 13696          VBUSDATA        <^X1E>,1                ;80
01              0339 13697          .BYTE   ^X01
                033A 13698          VBUSDATA        <^X1E>,0                ;81
01              033B 13699          .BYTE   ^X01
                033C 13700          VBUSDATA        <^X1E>,1                ;82
01              033D 13701          .BYTE   ^X01
                033E 13702          VBUSDATA        <^X1E>,0                ;83
01              033F 13703          .BYTE   ^X01
                0340 13704          VBUSDATA        <^X1E>,1                ;84
01              0341 13705          .BYTE   ^X01
                0342 13706          VBUSDATA        <^X1E>,1                ;85
01              0343 13707          .BYTE   ^X01
                0344 13708          VBUSDATA        <^X1E>,1                ;86
01              0345 13709          .BYTE   ^X01
                0346 13710          VBUSDATA        <^X1E>,1                ;87
01              0347 13711          .BYTE   ^X01
                0348 13712          VBUSDATA        <^X1E>,1                ;88
01              0349 13713          .BYTE   ^X01
                034A 13714          VBUSDATA        <^X1E>,0                ;89
01              034B 13715          .BYTE   ^X01
                034C 13716          VBUSDATA        <^X1E>,1                ;8A
01              034D 13717          .BYTE   ^X01
                034E 13718          VBUSDATA        <^X1E>,0                ;8B
```

```
01  034F 13719          .BYTE   ^X01
    0350 13720          VBUSDATA        <^X1E>,1                ;8C
01  0351 13721          .BYTE   ^X01
    0352 13722          VBUSDATA        <^X1E>,0                ;8D
01  0353 13723          .BYTE   ^X01
    0354 13724          VBUSDATA        <^X1E>,1                ;8E
01  0355 13725          .BYTE   ^X01
    0356 13726          VBUSDATA        <^X1E>,0                ;8F
01  0357 13727          .BYTE   ^X01
    0358 13728          VBUSDATA        <^X1E>,1                ;90
01  0359 13729          .BYTE   ^X01
    035A 13730          VBUSDATA        <^X1E>,1                ;91
01  035B 13731          .BYTE   ^X01
    035C 13732          VBUSDATA        <^X1E>,1                ;92
01  035D 13733          .BYTE   ^X01
    035E 13734          VBUSDATA        <^X1E>,1                ;93
01  035F 13735          .BYTE   ^X01
    0360 13736          VBUSDATA        <^X1E>,1                ;94
01  0361 13737          .BYTE   ^X01
    0362 13738          VBUSDATA        <^X1E>,1                ;95
01  0363 13739          .BYTE   ^X01
    0364 13740          VBUSDATA        <^X1E>,1                ;96
01  0365 13741          .BYTE   ^X01
    0366 13742          VBUSDATA        <^X1E>,1                ;97
01  0367 13743          .BYTE   ^X01
    0368 13744          VBUSDATA        <^X1E>,1                ;98
01  0369 13745          .BYTE   ^X01
    036A 13746          VBUSDATA        <^X1E>,1                ;99
01  036B 13747          .BYTE   ^X01
    036C 13748          VBUSDATA        <^X1E>,1                ;9A
01  036D 13749          .BYTE   ^X01
    036E 13750          VBUSDATA        <^X1E>,1                ;9B
01  036F 13751          .BYTE   ^X01
    0370 13752          VBUSDATA        <^X1E>,0                ;9C
01  0371 13753          .BYTE   ^X01
    0372 13754          VBUSDATA        <^X1E>,0                ;9D
01  0373 13755          .BYTE   ^X01
    0374 13756          VBUSDATA        <^X1E>,1                ;9E
01  0375 13757          .BYTE   ^X01
    0376 13758          VBUSDATA        <^X1E>,1                ;9F
01  0377 13759          .BYTE   ^X01
    0378 13760          VBUSDATA        <^X1E>,1                ;A0
01  0379 13761          .BYTE   ^X01
    037A 13762          VBUSDATA        <^X1E>,0                ;A1
01  037B 13763          .BYTE   ^X01
    037C 13764          VBUSDATA        <^X1E>,1                ;A2
01  037D 13765          .BYTE   ^X01
    037E 13766          VBUSDATA        <^X1E>,0                ;A3
01  037F 13767          .BYTE   ^X01
    0380 13768          VBUSDATA        <^X1E>,1                ;A4
01  0381 13769          .BYTE   ^X01
    0382 13770          VBUSDATA        <^X1E>,1                ;A5
01  0383 13771          .BYTE   ^X01
    0384 13772          VBUSDATA        <^X1E>,1                ;A6
01  0385 13773          .BYTE   ^X01
```

```
                    0386 13774          VBUSDATA        <^X1E>,1                ;A7
01                  0387 13775          .BYTE   ^X01
                    0388 13776          VBUSDATA        <^X1E>,1                ;A8
01                  0389 13777          .BYTE   ^X01
                    038A 13778          VBUSDATA        <^X1E>,0                ;A9
01                  038B 13779          .BYTE   ^X01
                    038C 13780          VBUSDATA        <^X1E>,1                ;AA
01                  038D 13781          .BYTE   ^X01
                    038E 13782          VBUSDATA        <^X1E>,0                ;AB
01                  038F 13783          .BYTE   ^X01
                    0390 13784          VBUSDATA        <^X1E>,1                ;AC
01                  0391 13785          .BYTE   ^X01
                    0392 13786          VBUSDATA        <^X1E>,0                ;AD
01                  0393 13787          .BYTE   ^X01
                    0394 13788          VBUSDATA        <^X1E>,1                ;AE
01                  0395 13789          .BYTE   ^X01
                    0396 13790          VBUSDATA        <^X1E>,0                ;AF
01                  0397 13791          .BYTE   ^X01
                    0398 13792          VBUSDATA        <^X1E>,1                ;B0
01                  0399 13793          .BYTE   ^X01
                    039A 13794          VBUSDATA        <^X1E>,1                ;B1
01                  039B 13795          .BYTE   ^X01
                    039C 13796          VBUSDATA        <^X1E>,1                ;B2
01                  039D 13797          .BYTE   ^X01
                    039E 13798          VBUSDATA        <^X1E>,1                ;B3
01                  039F 13799          .BYTE   ^X01
                    03A0 13800          VBUSDATA        <^X1E>,1                ;B4
01                  03A1 13801          .BYTE   ^X01
                    03A2 13802          VBUSDATA        <^X1E>,1                ;B5
01                  03A3 13803          .BYTE   ^X01
                    03A4 13804          VBUSDATA        <^X1E>,1                ;B6
01                  03A5 13805          .BYTE   ^X01
                    03A6 13806          VBUSDATA        <^X1E>,1                ;B7
01                  03A7 13807          .BYTE   ^X01
                    03A8 13808          VBUSDATA        <^X1E>,1                ;B8
01                  03A9 13809          .BYTE   ^X01
                    03AA 13810          VBUSDATA        <^X1E>,1                ;B9
01                  03AB 13811          .BYTE   ^X01
                    03AC 13812          VBUSDATA        <^X1E>,1                ;BA
01                  03AD 13813          .BYTE   ^X01
                    03AE 13814          VBUSDATA        <^X1E>,1                ;BB
01                  03AF 13815          .BYTE   ^X01
                    03B0 13816          VBUSDATA        <^X1E>,1                ;BC
01                  03B1 13817          .BYTE   ^X01
                    03B2 13818          VBUSDATA        <^X1E>,1                ;BD
01                  03B3 13819          .BYTE   ^X01
                    03B4 13820          VBUSDATA        <^X1E>,1                ;BE
01                  03B5 13821          .BYTE   ^X01
                    03B6 13822          VBUSDATA        <^X1E>,1                ;BF
01                  03B7 13823          .BYTE   ^X01
                    03B8 13824          VBUSDATA        <^X1E>,1                ;C0
01                  03B9 13825          .BYTE   ^X01
                    03BA 13826          VBUSDATA        <^X1E>,0                ;C1
01                  03BB 13827          .BYTE   ^X01
                    03BC 13828          VBUSDATA        <^X1E>,1                ;C2
```

```
                    01  03BD 13829          .BYTE   ^X01
                        03BE 13830          VBUSDATA        <^X1E>,0                ;C3
                    01  03BF 13831          .BYTE   ^X01
                        03C0 13832          VBUSDATA        <^X1E>,1                ;C4
                    01  03C1 13833          .BYTE   ^X01
                        03C2 13834          VBUSDATA        <^X1E>,1                ;C5
                    01  03C3 13835          .BYTE   ^X01
                        03C4 13836          VBUSDATA        <^X1E>,1                ;C6
                    01  03C5 13837          .BYTE   ^X01
                        03C6 13838          VBUSDATA        <^X1E>,1                ;C7
                    01  03C7 13839          .BYTE   ^X01
                        03C8 13840          VBUSDATA        <^X1E>,1                ;C8
                    01  03C9 13841          .BYTE   ^X01
                        03CA 13842          VBUSDATA        <^X1E>,0                ;C9
                    01  03CB 13843          .BYTE   ^X01
                        03CC 13844          VBUSDATA        <^X1E>,1                ;CA
                    01  03CD 13845          .BYTE   ^X01
                        03CE 13846          VBUSDATA        <^X1E>,0                ;CB
                    01  03CF 13847          .BYTE   ^X01
                        03D0 13848          VBUSDATA        <^X1E>,1                ;CC
                    01  03D1 13849          .BYTE   ^X01
                        03D2 13850          VBUSDATA        <^X1E>,0                ;CD
                    01  03D3 13851          .BYTE   ^X01
                        03D4 13852          VBUSDATA        <^X1E>,1                ;CE
                    01  03D5 13853          .BYTE   ^X01
                        03D6 13854          VBUSDATA        <^X1E>,0                ;CF
                    01  03D7 13855          .BYTE   ^X01
                        03D8 13856          VBUSDATA        <^X1E>,1                ;D0
                    01  03D9 13857          .BYTE   ^X01
                        03DA 13858          VBUSDATA        <^X1E>,1                ;D1
                    01  03DB 13859          .BYTE   ^X01
                        03DC 13860          VBUSDATA        <^X1E>,1                ;D2
                    01  03DD 13861          .BYTE   ^X01
                        03DE 13862          VBUSDATA        <^X1E>,1                ;D3
                    01  03DF 13863          .BYTE   ^X01
                        03E0 13864          VBUSDATA        <^X1E>,1                ;D4
                    01  03E1 13865          .BYTE   ^X01
                        03E2 13866          VBUSDATA        <^X1E>,1                ;D5
                    01  03E3 13867          .BYTE   ^X01
                        03E4 13868          VBUSDATA        <^X1E>,1                ;D6
                    01  03E5 13869          .BYTE   ^X01
                        03E6 13870          VBUSDATA        <^X1E>,1                ;D7
                    01  03E7 13871          .BYTE   ^X01
                        03E8 13872          VBUSDATA        <^X1E>,1                ;D8
                    01  03E9 13873          .BYTE   ^X01
                        03EA 13874          VBUSDATA        <^X1E>,1                ;D9
                    01  03EB 13875          .BYTE   ^X01
                        03EC 13876          VBUSDATA        <^X1E>,1                ;DA
                    01  03ED 13877          .BYTE   ^X01
                        03EE 13878          VBUSDATA        <^X1E>,1                ;DB
                    01  03EF 13879          .BYTE   ^X01
                        03F0 13880          VBUSDATA        <^X1E>,1                ;DC
                    01  03F1 13881          .BYTE   ^X01
                        03F2 13882          VBUSDATA        <^X1E>,1                ;DD
                    01  03F3 13883          .BYTE   ^X01
```

```
        03F4 13884      VBUSDATA        <^X1E>,1                ;DE
   01   03F5 13885      .BYTE   ^X01
        03F6 13886      VBUSDATA        <^X1E>,1                ;DF
   01   03F7 13887      .BYTE   ^X01
        03F8 13888      VBUSDATA        <^X1E>,1                ;E0
   01   03F9 13889      .BYTE   ^X01
        03FA 13890      VBUSDATA        <^X1E>,1                ;E1
   01   03FB 13891      .BYTE   ^X01
        03FC 13892      VBUSDATA        <^X1E>,1                ;E2
   01   03FD 13893      .BYTE   ^X01
        03FE 13894      VBUSDATA        <^X1E>,1                ;E3
   01   03FF 13895      .BYTE   ^X01
        0400 13896      VBUSDATA        <^X1E>,1                ;E4
   01   0401 13897      .BYTE   ^X01
        0402 13898      VBUSDATA        <^X1E>,1                ;E5
   01   0403 13899      .BYTE   ^X01
        0404 13900      VBUSDATA        <^X1E>,1                ;E6
   01   0405 13901      .BYTE   ^X01
        0406 13902      VBUSDATA        <^X1E>,1                ;E7
   01   0407 13903      .BYTE   ^X01
        0408 13904      VBUSDATA        <^X1E>,1                ;E8
   01   0409 13905      .BYTE   ^X01
        040A 13906      VBUSDATA        <^X1E>,1                ;E9
   01   040B 13907      .BYTE   ^X01
        040C 13908      VBUSDATA        <^X1E>,0                ;EA
   01   040D 13909      .BYTE   ^X01
        040E 13910      VBUSDATA        <^X1E>,0                ;EB
   01   040F 13911      .BYTE   ^X01
        0410 13912      VBUSDATA        <^X1E>,0                ;EC
   01   0411 13913      .BYTE   ^X01
        0412 13914      VBUSDATA        <^X1E>,0                ;ED
   01   0413 13915      .BYTE   ^X01
        0414 13916      VBUSDATA        <^X1E>,0                ;EE
   01   0415 13917      .BYTE   ^X01
        0416 13918      VBUSDATA        <^X1E>,0                ;EF
   01   0417 13919      .BYTE   ^X01
        0418 13920      VBUSDATA        <^X1E>,1                ;F0
   01   0419 13921      .BYTE   ^X01
        041A 13922      VBUSDATA        <^X1E>,0                ;F1
   01   041B 13923      .BYTE   ^X01
        041C 13924      VBUSDATA        <^X1E>,1                ;F2
   01   041D 13925      .BYTE   ^X01
        041E 13926      VBUSDATA        <^X1E>,1                ;F3
   01   041F 13927      .BYTE   ^X01
        0420 13928      VBUSDATA        <^X1E>,1                ;F4
   01   0421 13929      .BYTE   ^X01
        0422 13930      VBUSDATA        <^X1E>,1                ;F5
   01   0423 13931      .BYTE   ^X01
        0424 13932      VBUSDATA        <^X1E>,1                ;F6
   01   0425 13933      .BYTE   ^X01
        0426 13934      VBUSDATA        <^X1E>,1                ;F7
   01   0427 13935      .BYTE   ^X01
        0428 13936      VBUSDATA        <^X1E>,1                ;F8
   01   0429 13937      .BYTE   ^X01
        042A 13938      VBUSDATA        <^X1E>,1                ;F9
```

```
01  042B 13939          .BYTE   ^X01
    042C 13940          VBUSDATA        <^X1E>,1                ;FA
01  042D 13941          .BYTE   ^X01
    042E 13942          VBUSDATA        <^X1E>,1                ;FB
01  042F 13943          .BYTE   ^X01
    0430 13944          VBUSDATA        <^X1E>,1                ;FC
01  0431 13945          .BYTE   ^X01
    0432 13946          VBUSDATA        <^X1E>,1                ;FD
01  0433 13947          .BYTE   ^X01
    0434 13948          VBUSDATA        <^X1E>,1                ;FE
01  0435 13949          .BYTE   ^X01
    0436 13950          VBUSDATA        <^X1E>,1                ;FF
    0437 13951
    0437 13952
    0437 13953 ;
    0437 13954 ; Expected VBUS values for Subtest 3
    0437 13955 ;
    0437 13956
01  0437 13957 31$:     .BYTE   ^X01
    0438 13958          VBUSDATA        <^X1E>,1                ;00
01  0439 13959          .BYTE   ^X01
    043A 13960          VBUSDATA        <^X1E>,1                ;01
01  043B 13961          .BYTE   ^X01
    043C 13962          VBUSDATA        <^X1E>,1                ;02
01  043D 13963          .BYTE   ^X01
    043E 13964          VBUSDATA        <^X1E>,1                ;03
01  043F 13965          .BYTE   ^X01
    0440 13966          VBUSDATA        <^X1E>,1                ;04
01  0441 13967          .BYTE   ^X01
    0442 13968          VBUSDATA        <^X1E>,1                ;05
01  0443 13969          .BYTE   ^X01
    0444 13970          VBUSDATA        <^X1E>,1                ;06
01  0445 13971          .BYTE   ^X01
    0446 13972          VBUSDATA        <^X1E>,1                ;07
01  0447 13973          .BYTE   ^X01
    0448 13974          VBUSDATA        <^X1E>,1                ;08
01  0449 13975          .BYTE   ^X01
    044A 13976          VBUSDATA        <^X1E>,1                ;09
01  044B 13977          .BYTE   ^X01
    044C 13978          VBUSDATA        <^X1E>,1                ;0A
01  044D 13979          .BYTE   ^X01
    044E 13980          VBUSDATA        <^X1E>,1                ;0B
01  044F 13981          .BYTE   ^X01
    0450 13982          VBUSDATA        <^X1E>,1                ;0C
01  0451 13983          .BYTE   ^X01
    0452 13984          VBUSDATA        <^X1E>,1                ;0D
01  0453 13985          .BYTE   ^X01
    0454 13986          VBUSDATA        <^X1E>,1                ;0E
01  0455 13987          .BYTE   ^X01
    0456 13988          VBUSDATA        <^X1E>,1                ;0F
01  0457 13989          .BYTE   ^X01
    0458 13990          VBUSDATA        <^X1E>,1                ;10
01  0459 13991          .BYTE   ^X01
    045A 13992          VBUSDATA        <^X1E>,1                ;11
01  045B 13993          .BYTE   ^X01
```

```
                045C 13994          VBUSDATA        <^X1E>,1                ;12
            01  045D 13995          .BYTE   ^X01
                045E 13996          VBUSDATA        <^X1E>,1                ;13
            01  045F 13997          .BYTE   ^X01
                0460 13998          VBUSDATA        <^X1E>,1                ;14
            01  0461 13999          .BYTE   ^X01
                0462 14000          VBUSDATA        <^X1E>,1                ;15
            01  0463 14001          .BYTE   ^X01
                0464 14002          VBUSDATA        <^X1E>,1                ;16
            01  0465 14003          .BYTE   ^X01
                0466 14004          VBUSDATA        <^X1E>,1                ;17
            01  0467 14005          .BYTE   ^X01
                0468 14006          VBUSDATA        <^X1E>,1                ;18
            01  0469 14007          .BYTE   ^X01
                046A 14008          VBUSDATA        <^X1E>,1                ;19
            01  046B 14009          .BYTE   ^X01
                046C 14010          VBUSDATA        <^X1E>,1                ;1A
            01  046D 14011          .BYTE   ^X01
                046E 14012          VBUSDATA        <^X1E>,1                ;1B
            01  046F 14013          .BYTE   ^X01
                0470 14014          VBUSDATA        <^X1E>,1                ;1C
            01  0471 14015          .BYTE   ^X01
                0472 14016          VBUSDATA        <^X1E>,1                ;1D
            01  0473 14017          .BYTE   ^X01
                0474 14018          VBUSDATA        <^X1E>,1                ;1E
            01  0475 14019          .BYTE   ^X01
                0476 14020          VBUSDATA        <^X1E>,1                ;1F
            01  0477 14021          .BYTE   ^X01
                0478 14022          VBUSDATA        <^X1E>,1                ;20
            01  0479 14023          .BYTE   ^X01
                047A 14024          VBUSDATA        <^X1E>,1                ;21
            01  047B 14025          .BYTE   ^X01
                047C 14026          VBUSDATA        <^X1E>,1                ;22
            01  047D 14027          .BYTE   ^X01
                047E 14028          VBUSDATA        <^X1E>,1                ;23
            01  047F 14029          .BYTE   ^X01
                0480 14030          VBUSDATA        <^X1E>,1                ;24
            01  0481 14031          .BYTE   ^X01
                0482 14032          VBUSDATA        <^X1E>,1                ;25
            01  0483 14033          .BYTE   ^X01
                0484 14034          VBUSDATA        <^X1E>,1                ;26
            01  0485 14035          .BYTE   ^X01
                0486 14036          VBUSDATA        <^X1E>,1                ;27
            01  0487 14037          .BYTE   ^X01
                0488 14038          VBUSDATA        <^X1E>,1                ;28
            01  0489 14039          .BYTE   ^X01
                048A 14040          VBUSDATA        <^X1E>,1                ;29
            01  048B 14041          .BYTE   ^X01
                048C 14042          VBUSDATA        <^X1E>,1                ;2A
            01  048D 14043          .BYTE   ^X01
                048E 14044          VBUSDATA        <^X1E>,1                ;2B
            01  048F 14045          .BYTE   ^X01
                0490 14046          VBUSDATA        <^X1E>,1                ;2C
            01  0491 14047          .BYTE   ^X01
                0492 14048          VBUSDATA        <^X1E>,1                ;2D
```

```
01  0493 14049          .BYTE   ^X01
    0494 14050          VBUSDATA        <^X1E>,1                ;2E
01  0495 14051          .BYTE   ^X01
    0496 14052          VBUSDATA        <^X1E>,1                ;2F
01  0497 14053          .BYTE   ^X01
    0498 14054          VBUSDATA        <^X1E>,1                ;30
01  0499 14055          .BYTE   ^X01
    049A 14056          VBUSDATA        <^X1E>,1                ;31
01  049B 14057          .BYTE   ^X01
    049C 14058          VBUSDATA        <^X1E>,1                ;32
01  049D 14059          .BYTE   ^X01
    049E 14060          VBUSDATA        <^X1E>,1                ;33
01  049F 14061          .BYTE   ^X01
    04A0 14062          VBUSDATA        <^X1E>,1                ;34
01  04A1 14063          .BYTE   ^X01
    04A2 14064          VBUSDATA        <^X1E>,1                ;35
01  04A3 14065          .BYTE   ^X01
    04A4 14066          VBUSDATA        <^X1E>,1                ;36
01  04A5 14067          .BYTE   ^X01
    04A6 14068          VBUSDATA        <^X1E>,1                ;37
01  04A7 14069          .BYTE   ^X01
    04A8 14070          VBUSDATA        <^X1E>,1                ;38
01  04A9 14071          .BYTE   ^X01
    04AA 14072          VBUSDATA        <^X1E>,1                ;39
01  04AB 14073          .BYTE   ^X01
    04AC 14074          VBUSDATA        <^X1E>,1                ;3A
01  04AD 14075          .BYTE   ^X01
    04AE 14076          VBUSDATA        <^X1E>,1                ;3B
01  04AF 14077          .BYTE   ^X01
    04B0 14078          VBUSDATA        <^X1E>,1                ;3C
01  04B1 14079          .BYTE   ^X01
    04B2 14080          VBUSDATA        <^X1E>,0                ;3D
01  04B3 14081          .BYTE   ^X01
    04B4 14082          VBUSDATA        <^X1E>,1                ;3E
01  04B5 14083          .BYTE   ^X01
    04B6 14084          VBUSDATA        <^X1E>,1                ;3F
01  04B7 14085          .BYTE   ^X01
    04B8 14086          VBUSDATA        <^X1E>,1                ;40
01  04B9 14087          .BYTE   ^X01
    04BA 14088          VBUSDATA        <^X1E>,1                ;41
01  04BB 14089          .BYTE   ^X01
    04BC 14090          VBUSDATA        <^X1E>,1                ;42
01  04BD 14091          .BYTE   ^X01
    04BE 14092          VBUSDATA        <^X1E>,1                ;43
01  04BF 14093          .BYTE   ^X01
    04C0 14094          VBUSDATA        <^X1E>,1                ;44
01  04C1 14095          .BYTE   ^X01
    04C2 14096          VBUSDATA        <^X1E>,1                ;45
01  04C3 14097          .BYTE   ^X01
    04C4 14098          VBUSDATA        <^X1E>,1                ;46
01  04C5 14099          .BYTE   ^X01
    04C6 14100          VBUSDATA        <^X1E>,1                ;47
01  04C7 14101          .BYTE   ^X01
    04C8 14102          VBUSDATA        <^X1E>,1                ;48
01  04C9 14103          .BYTE   ^X01
```

```
        04CA 14104        VBUSDATA        <^X1E>,1                ;49
   01   04CB 14105        .BYTE   ^X01
        04CC 14106        VBUSDATA        <^X1E>,1                ;4A
   01   04CD 14107        .BYTE   ^X01
        04CE 14108        VBUSDATA        <^X1E>,1                ;4B
   01   04CF 14109        .BYTE   ^X01
        04D0 14110        VBUSDATA        <^X1E>,1                ;4C
   01   04D1 14111        .BYTE   ^X01
        04D2 14112        VBUSDATA        <^X1E>,1                ;4D
   01   04D3 14113        .BYTE   ^X01
        04D4 14114        VBUSDATA        <^X1E>,1                ;4E
   01   04D5 14115        .BYTE   ^X01
        04D6 14116        VBUSDATA        <^X1E>,1                ;4F
   01   04D7 14117        .BYTE   ^X01
        04D8 14118        VBUSDATA        <^X1E>,1                ;50
   01   04D9 14119        .BYTE   ^X01
        04DA 14120        VBUSDATA        <^X1E>,1                ;51
   01   04DB 14121        .BYTE   ^X01
        04DC 14122        VBUSDATA        <^X1E>,1                ;52
   01   04DD 14123        .BYTE   ^X01
        04DE 14124        VBUSDATA        <^X1E>,1                ;53
   01   04DF 14125        .BYTE   ^X01
        04E0 14126        VBUSDATA        <^X1E>,1                ;54
   01   04E1 14127        .BYTE   ^X01
        04E2 14128        VBUSDATA        <^X1E>,1                ;55
   01   04E3 14129        .BYTE   ^X01
        04E4 14130        VBUSDATA        <^X1E>,1                ;56
   01   04E5 14131        .BYTE   ^X01
        04E6 14132        VBUSDATA        <^X1E>,1                ;57
   01   04E7 14133        .BYTE   ^X01
        04E8 14134        VBUSDATA        <^X1E>,1                ;58
   01   04E9 14135        .BYTE   ^X01
        04EA 14136        VBUSDATA        <^X1E>,1                ;59
   01   04EB 14137        .BYTE   ^X01
        04EC 14138        VBUSDATA        <^X1E>,1                ;5A
   01   04ED 14139        .BYTE   ^X01
        04EE 14140        VBUSDATA        <^X1E>,1                ;5B
   01   04EF 14141        .BYTE   ^X01
        04F0 14142        VBUSDATA        <^X1E>,1                ;5C
   01   04F1 14143        .BYTE   ^X01
        04F2 14144        VBUSDATA        <^X1E>,1                ;5D
   01   04F3 14145        .BYTE   ^X01
        04F4 14146        VBUSDATA        <^X1E>,1                ;5E
   01   04F5 14147        .BYTE   ^X01
        04F6 14148        VBUSDATA        <^X1E>,1                ;5F
   01   04F7 14149        .BYTE   ^X01
        04F8 14150        VBUSDATA        <^X1E>,1                ;60
   01   04F9 14151        .BYTE   ^X01
        04FA 14152        VBUSDATA        <^X1E>,1                ;61
   01   04FB 14153        .BYTE   ^X01
        04FC 14154        VBUSDATA        <^X1E>,1                ;62
   01   04FD 14155        .BYTE   ^X01
        04FE 14156        VBUSDATA        <^X1E>,1                ;63
   01   04FF 14157        .BYTE   ^X01
        0500 14158        VBUSDATA        <^X1E>,1                ;64
```

```
01  0501 14159        .BYTE   ^X01
    0502 14160        VBUSDATA        <^X1E>,1                ;65
01  0503 14161        .BYTE   ^X01
    0504 14162        VBUSDATA        <^X1E>,1                ;66
01  0505 14163        .BYTE   ^X01
    0506 14164        VBUSDATA        <^X1E>,1                ;67
01  0507 14165        .BYTE   ^X01
    0508 14166        VBUSDATA        <^X1E>,1                ;68
01  0509 14167        .BYTE   ^X01
    050A 14168        VBUSDATA        <^X1E>,1                ;69
01  050B 14169        .BYTE   ^X01
    050C 14170        VBUSDATA        <^X1E>,1                ;6A
01  050D 14171        .BYTE   ^X01
    050E 14172        VBUSDATA        <^X1E>,1                ;6B
01  050F 14173        .BYTE   ^X01
    0510 14174        VBUSDATA        <^X1E>,1                ;6C
01  0511 14175        .BYTE   ^X01
    0512 14176        VBUSDATA        <^X1E>,1                ;6D
01  0513 14177        .BYTE   ^X01
    0514 14178        VBUSDATA        <^X1E>,1                ;6E
01  0515 14179        .BYTE   ^X01
    0516 14180        VBUSDATA        <^X1E>,1                ;6F
01  0517 14181        .BYTE   ^X01
    0518 14182        VBUSDATA        <^X1E>,1                ;70
01  0519 14183        .BYTE   ^X01
    051A 14184        VBUSDATA        <^X1E>,1                ;71
01  051B 14185        .BYTE   ^X01
    051C 14186        VBUSDATA        <^X1E>,1                ;72
01  051D 14187        .BYTE   ^X01
    051E 14188        VBUSDATA        <^X1E>,1                ;73
01  051F 14189        .BYTE   ^X01
    0520 14190        VBUSDATA        <^X1E>,1                ;74
01  0521 14191        .BYTE   ^X01
    0522 14192        VBUSDATA        <^X1E>,1                ;75
01  0523 14193        .BYTE   ^X01
    0524 14194        VBUSDATA        <^X1E>,1                ;76
01  0525 14195        .BYTE   ^X01
    0526 14196        VBUSDATA        <^X1E>,1                ;77
01  0527 14197        .BYTE   ^X01
    0528 14198        VBUSDATA        <^X1E>,1                ;78
01  0529 14199        .BYTE   ^X01
    052A 14200        VBUSDATA        <^X1E>,1                ;79
01  052B 14201        .BYTE   ^X01
    052C 14202        VBUSDATA        <^X1E>,1                ;7A
01  052D 14203        .BYTE   ^X01
    052E 14204        VBUSDATA        <^X1E>,1                ;7B
01  052F 14205        .BYTE   ^X01
    0530 14206        VBUSDATA        <^X1E>,1                ;7C
01  0531 14207        .BYTE   ^X01
    0532 14208        VBUSDATA        <^X1E>,1                ;7D
01  0533 14209        .BYTE   ^X01
    0534 14210        VBUSDATA        <^X1E>,1                ;7E
01  0535 14211        .BYTE   ^X01
    0536 14212        VBUSDATA        <^X1E>,1                ;7F
01  0537 14213        .BYTE   ^X01
```

```
                0538 14214          VBUSDATA        <^X1E>,1                ;80
01              0539 14215          .BYTE   ^X01
                053A 14216          VBUSDATA        <^X1E>,0                ;81
01              053B 14217          .BYTE   ^X01
                053C 14218          VBUSDATA        <^X1E>,1                ;82
01              053D 14219          .BYTE   ^X01
                053E 14220          VBUSDATA        <^X1E>,0                ;83
01              053F 14221          .BYTE   ^X01
                0540 14222          VBUSDATA        <^X1E>,1                ;84
01              0541 14223          .BYTE   ^X01
                0542 14224          VBUSDATA        <^X1E>,1                ;85
01              0543 14225          .BYTE   ^X01
                0544 14226          VBUSDATA        <^X1E>,1                ;86
01              0545 14227          .BYTE   ^X01
                0546 14228          VBUSDATA        <^X1E>,1                ;87
01              0547 14229          .BYTE   ^X01
                0548 14230          VBUSDATA        <^X1E>,1                ;88
01              0549 14231          .BYTE   ^X01
                054A 14232          VBUSDATA        <^X1E>,0                ;89
01              054B 14233          .BYTE   ^X01
                054C 14234          VBUSDATA        <^X1E>,1                ;8A
01              054D 14235          .BYTE   ^X01
                054E 14236          VBUSDATA        <^X1E>,0                ;8B
01              054F 14237          .BYTE   ^X01
                0550 14238          VBUSDATA        <^X1E>,1                ;8C
01              0551 14239          .BYTE   ^X01
                0552 14240          VBUSDATA        <^X1E>,0                ;8D
01              0553 14241          .BYTE   ^X01
                0554 14242          VBUSDATA        <^X1E>,1                ;8E
01              0555 14243          .BYTE   ^X01
                0556 14244          VBUSDATA        <^X1E>,0                ;8F
01              0557 14245          .BYTE   ^X01
                0558 14246          VBUSDATA        <^X1E>,1                ;90
01              0559 14247          .BYTE   ^X01
                055A 14248          VBUSDATA        <^X1E>,1                ;91
01              055B 14249          .BYTE   ^X01
                055C 14250          VBUSDATA        <^X1E>,1                ;92
01              055D 14251          .BYTE   ^X01
                055E 14252          VBUSDATA        <^X1E>,1                ;93
01              055F 14253          .BYTE   ^X01
                0560 14254          VBUSDATA        <^X1E>,1                ;94
01              0561 14255          .BYTE   ^X01
                0562 14256          VBUSDATA        <^X1E>,1                ;95
01              0563 14257          .BYTE   ^X01
                0564 14258          VBUSDATA        <^X1E>,1                ;96
01              0565 14259          .BYTE   ^X01
                0566 14260          VBUSDATA        <^X1E>,1                ;97
01              0567 14261          .BYTE   ^X01
                0568 14262          VBUSDATA        <^X1E>,1                ;98
01              0569 14263          .BYTE   ^X01
                056A 14264          VBUSDATA        <^X1E>,1                ;99
01              056B 14265          .BYTE   ^X01
                056C 14266          VBUSDATA        <^X1E>,1                ;9A
01              056D 14267          .BYTE   ^X01
                056E 14268          VBUSDATA        <^X1E>,1                ;9B
```

```
01   056F 14269          .BYTE    ^X01
     0570 14270          VBUSDATA         <^X1E>,0                ;9C
01   0571 14271          .BYTE    ^X01
     0572 14272          VBUSDATA         <^X1E>,0                ;9D
01   0573 14273          .BYTE    ^X01
     0574 14274          VBUSDATA         <^X1E>,1                ;9E
01   0575 14275          .BYTE    ^X01
     0576 14276          VBUSDATA         <^X1E>,1                ;9F
01   0577 14277          .BYTE    ^X01
     0578 14278          VBUSDATA         <^X1E>,1                ;A0
01   0579 14279          .BYTE    ^X01
     057A 14280          VBUSDATA         <^X1E>,0                ;A1
01   057B 14281          .BYTE    ^X01
     057C 14282          VBUSDATA         <^X1E>,1                ;A2
01   057D 14283          .BYTE    ^X01
     057E 14284          VBUSDATA         <^X1E>,0                ;A3
01   057F 14285          .BYTE    ^X01
     0580 14286          VBUSDATA         <^X1E>,1                ;A4
01   0581 14287          .BYTE    ^X01
     0582 14288          VBUSDATA         <^X1E>,1                ;A5
01   0583 14289          .BYTE    ^X01
     0584 14290          VBUSDATA         <^X1E>,1                ;A6
01   0585 14291          .BYTE    ^X01
     0586 14292          VBUSDATA         <^X1E>,1                ;A7
01   0587 14293          .BYTE    ^X01
     0588 14294          VBUSDATA         <^X1E>,1                ;A8
01   0589 14295          .BYTE    ^X01
     058A 14296          VBUSDATA         <^X1E>,0                ;A9
01   058B 14297          .BYTE    ^X01
     058C 14298          VBUSDATA         <^X1E>,1                ;AA
01   058D 14299          .BYTE    ^X01
     058E 14300          VBUSDATA         <^X1E>,0                ;AB
01   058F 14301          .BYTE    ^X01
     0590 14302          VBUSDATA         <^X1E>,1                ;AC
01   0591 14303          .BYTE    ^X01
     0592 14304          VBUSDATA         <^X1E>,0                ;AD
01   0593 14305          .BYTE    ^X01
     0594 14306          VBUSDATA         <^X1E>,1                ;AE
01   0595 14307          .BYTE    ^X01
     0596 14308          VBUSDATA         <^X1E>,0                ;AF
01   0597 14309          .BYTE    ^X01
     0598 14310          VBUSDATA         <^X1E>,1                ;B0
01   0599 14311          .BYTE    ^X01
     059A 14312          VBUSDATA         <^X1E>,1                ;B1
01   059B 14313          .BYTE    ^X01
     059C 14314          VBUSDATA         <^X1E>,1                ;B2
01   059D 14315          .BYTE    ^X01
     059E 14316          VBUSDATA         <^X1E>,1                ;B3
01   059F 14317          .BYTE    ^X01
     05A0 14318          VBUSDATA         <^X1E>,1                ;B4
01   05A1 14319          .BYTE    ^X01
     05A2 14320          VBUSDATA         <^X1E>,1                ;B5
01   05A3 14321          .BYTE    ^X01
     05A4 14322          VBUSDATA         <^X1E>,1                ;B6
01   05A5 14323          .BYTE    ^X01
```

```
            05A6 14324          VBUSDATA        <^X1E>,1                ;B7
 01         05A7 14325          .BYTE   ^X01
            05A8 14326          VBUSDATA        <^X1E>,1                ;B8
 01         05A9 14327          .BYTE   ^X01
            05AA 14328          VBUSDATA        <^X1E>,1                ;B9
 01         05AB 14329          .BYTE   ^X01
            05AC 14330          VBUSDATA        <^X1E>,1                ;BA
 01         05AD 14331          .BYTE   ^X01
            05AE 14332          VBUSDATA        <^X1E>,1                ;BB
 01         05AF 14333          .BYTE   ^X01
            05B0 14334          VBUSDATA        <^X1E>,1                ;BC
 01         05B1 14335          .BYTE   ^X01
            05B2 14336          VBUSDATA        <^X1E>,1                ;BD
 01         05B3 14337          .BYTE   ^X01
            05B4 14338          VBUSDATA        <^X1E>,1                ;BE
 01         05B5 14339          .BYTE   ^X01
            05B6 14340          VBUSDATA        <^X1E>,1                ;BF
 01         05B7 14341          .BYTE   ^X01
            05B8 14342          VBUSDATA        <^X1E>,1                ;C0
 01         05B9 14343          .BYTE   ^X01
            05BA 14344          VBUSDATA        <^X1E>,0                ;C1
 01         05BB 14345          .BYTE   ^X01
            05BC 14346          VBUSDATA        <^X1E>,1                ;C2
 01         05BD 14347          .BYTE   ^X01
            05BE 14348          VBUSDATA        <^X1E>,0                ;C3
 01         05BF 14349          .BYTE   ^X01
            05C0 14350          VBUSDATA        <^X1E>,1                ;C4
 01         05C1 14351          .BYTE   ^X01
            05C2 14352          VBUSDATA        <^X1E>,1                ;C5
 01         05C3 14353          .BYTE   ^X01
            05C4 14354          VBUSDATA        <^X1E>,1                ;C6
 01         05C5 14355          .BYTE   ^X01
            05C6 14356          VBUSDATA        <^X1E>,1                ;C7
 01         05C7 14357          .BYTE   ^X01
            05C8 14358          VBUSDATA        <^X1E>,1                ;C8
 01         05C9 14359          .BYTE   ^X01
            05CA 14360          VBUSDATA        <^X1E>,0                ;C9
 01         05CB 14361          .BYTE   ^X01
            05CC 14362          VBUSDATA        <^X1E>,1                ;CA
 01         05CD 14363          .BYTE   ^X01
            05CE 14364          VBUSDATA        <^X1E>,0                ;CB
 01         05CF 14365          .BYTE   ^X01
            05D0 14366          VBUSDATA        <^X1E>,1                ;CC
 01         05D1 14367          .BYTE   ^X01
            05D2 14368          VBUSDATA        <^X1E>,0                ;CD
 01         05D3 14369          .BYTE   ^X01
            05D4 14370          VBUSDATA        <^X1E>,1                ;CE
 01         05D5 14371          .BYTE   ^X01
            05D6 14372          VBUSDATA        <^X1E>,0                ;CF
 01         05D7 14373          .BYTE   ^X01
            05D8 14374          VBUSDATA        <^X1E>,1                ;D0
 01         05D9 14375          .BYTE   ^X01
            05DA 14376          VBUSDATA        <^X1E>,1                ;D1
 01         05DB 14377          .BYTE   ^X01
            05DC 14378          VBUSDATA        <^X1E>,1                ;D2
```

```
01  05DD 14379          .BYTE   ^X01
    05DE 14380          VBUSDATA        <^X1E>,1                ;D3
01  05DF 14381          .BYTE   ^X01
    05E0 14382          VBUSDATA        <^X1E>,1                ;D4
01  05E1 14383          .BYTE   ^X01
    05E2 14384          VBUSDATA        <^X1E>,1                ;D5
01  05E3 14385          .BYTE   ^X01
    05E4 14386          VBUSDATA        <^X1E>,1                ;D6
01  05E5 14387          .BYTE   ^X01
    05E6 14388          VBUSDATA        <^X1E>,1                ;D7
01  05E7 14389          .BYTE   ^X01
    05E8 14390          VBUSDATA        <^X1E>,1                ;D8
01  05E9 14391          .BYTE   ^X01
    05EA 14392          VBUSDATA        <^X1E>,1                ;D9
01  05EB 14393          .BYTE   ^X01
    05EC 14394          VBUSDATA        <^X1E>,1                ;DA
01  05ED 14395          .BYTE   ^X01
    05EE 14396          VBUSDATA        <^X1E>,1                ;DB
01  05EF 14397          .BYTE   ^X01
    05F0 14398          VBUSDATA        <^X1E>,1                ;DC
01  05F1 14399          .BYTE   ^X01
    05F2 14400          VBUSDATA        <^X1E>,1                ;DD
01  05F3 14401          .BYTE   ^X01
    05F4 14402          VBUSDATA        <^X1E>,1                ;DE
01  05F5 14403          .BYTE   ^X01
    05F6 14404          VBUSDATA        <^X1E>,1                ;DF
01  05F7 14405          .BYTE   ^X01
    05F8 14406          VBUSDATA        <^X1E>,1                ;E0
01  05F9 14407          .BYTE   ^X01
    05FA 14408          VBUSDATA        <^X1E>,1                ;E1
01  05FB 14409          .BYTE   ^X01
    05FC 14410          VBUSDATA        <^X1E>,1                ;E2
01  05FD 14411          .BYTE   ^X01
    05FE 14412          VBUSDATA        <^X1E>,1                ;E3
01  05FF 14413          .BYTE   ^X01
    0600 14414          VBUSDATA        <^X1E>,1                ;E4
01  0601 14415          .BYTE   ^X01
    0602 14416          VBUSDATA        <^X1E>,1                ;E5
01  0603 14417          .BYTE   ^X01
    0604 14418          VBUSDATA        <^X1E>,1                ;E6
01  0605 14419          .BYTE   ^X01
    0606 14420          VBUSDATA        <^X1E>,1                ;E7
01  0607 14421          .BYTE   ^X01
    0608 14422          VBUSDATA        <^X1E>,1                ;E8
01  0609 14423          .BYTE   ^X01
    060A 14424          VBUSDATA        <^X1E>,1                ;E9
01  060B 14425          .BYTE   ^X01
    060C 14426          VBUSDATA        <^X1E>,0                ;EA
01  060D 14427          .BYTE   ^X01
    060E 14428          VBUSDATA        <^X1E>,0                ;EB
01  060F 14429          .BYTE   ^X01
    0610 14430          VBUSDATA        <^X1E>,0                ;EC
01  0611 14431          .BYTE   ^X01
    0612 14432          VBUSDATA        <^X1E>,0                ;ED
01  0613 14433          .BYTE   ^X01
```

```
                        0614 14434          VBUSDATA        <^X1E>,0                ;EE
                  01    0615 14435          .BYTE   ^X01
                        0616 14436          VBUSDATA        <^X1E>,0                ;EF
                  01    0617 14437          .BYTE   ^X01
                        0618 14438          VBUSDATA        <^X1E>,1                ;F0
                  01    0619 14439          .BYTE   ^X01
                        061A 14440          VBUSDATA        <^X1E>,0                ;F1
                  01    061B 14441          .BYTE   ^X01
                        061C 14442          VBUSDATA        <^X1E>,1                ;F2
                  01    061D 14443          .BYTE   ^X01
                        061E 14444          VBUSDATA        <^X1E>,1                ;F3
                  01    061F 14445          .BYTE   ^X01
                        0620 14446          VBUSDATA        <^X1E>,1                ;F4
                  01    0621 14447          .BYTE   ^X01
                        0622 14448          VBUSDATA        <^X1E>,1                ;F5
                  01    0623 14449          .BYTE   ^X01
                        0624 14450          VBUSDATA        <^X1E>,1                ;F6
                  01    0625 14451          .BYTE   ^X01
                        0626 14452          VBUSDATA        <^X1E>,1                ;F7
                  01    0627 14453          .BYTE   ^X01
                        0628 14454          VBUSDATA        <^X1E>,1                ;F8
                  01    0629 14455          .BYTE   ^X01
                        062A 14456          VBUSDATA        <^X1E>,1                ;F9
                  01    062B 14457          .BYTE   ^X01
                        062C 14458          VBUSDATA        <^X1E>,1                ;FA
                  01    062D 14459          .BYTE   ^X01
                        062E 14460          VBUSDATA        <^X1E>,1                ;FB
                  01    062F 14461          .BYTE   ^X01
                        0630 14462          VBUSDATA        <^X1E>,1                ;FC
                  01    0631 14463          .BYTE   ^X01
                        0632 14464          VBUSDATA        <^X1E>,1                ;FD
                  01    0633 14465          .BYTE   ^X01
                        0634 14466          VBUSDATA        <^X1E>,1                ;FE
                  01    0635 14467          .BYTE   ^X01
                        0636 14468          VBUSDATA        <^X1E>,1                ;FF
                        0637 14469
                        0637 14470 BEGIN_CPU_DATA
                        0002 14471
                        0002 14472 DCS_DATA
                        0001 14473
U 00, 7700,C800,0364,0300,0470,1801     0001 14474 ;% 00    NOP                             ; get cpu up to speed
U 01, 5700,C840,0364,0308,0470,1802     000C 14478 ;% 01    R[TEMP2]_RDM                    ; load opcode in R[TEMP2]
U 02, 7700,F800,56D7,57FF,8470,1803     0017 14482 ;% 02    LONLIT_[52515000]               ; load opspecs in lonlit reg.
U 03, 7700,8800,5BE5,03D4,0470,1804     0022 14486 ;% 03    Q_R[LONLIT]                     ; move opspecs to Q register
U 04, 7700,0842,00A4,0308,0470,1805     002D 14490 ;% 04    R[TEMP2]_M[TEMP2].OR.Q          ; OR opcode and opspecs
U 05, 7700,4868,5BE4,0308,0470,1806     0038 14494 ;% 05    M[TEMP8]_R[TEMP2]               ; save in M[TEMP8] for error
U 06, 7700,C800,5BE4,03D8,4A70,1807     0043 14498 ;% 06    VA_R[ZERO]                      ; zero the VA register
U 07, 7701,C800,5BE4,0208,5590,1808     004E 14502 ;% 07    WRITE.LONG R[TEMP2]             ; write data for XB0
U 08, 7700,0800,5BE4,0308,5470,1809     0059 14506 ;% 08    WDR_UNROT(R[TEMP2])             ; validate cache location 0
U 09, 7700,C800,0364,0300,4470,180A     0064 14510 ;% 09    VA_VA+4                         ; update the VA by 4
U 0A, 7700,3800,54D5,55D6,0470,180B     006F 14514 ;% 0A    LONLIT_[56555453]               ; load opspecs in lonlit reg.
U 0B, 7700,4869,5BE4,03D4,0470,180C     007A 14518 ;% 0B    M[TEMP9]_R[LONLIT]              ; save in M[TEMP9] for error
U 0C, 7701,8800,5BE4,02D4,5590,180D     0085 14522 ;% 0C    WRITE.LONG R[LONLIT]            ; write data for XB1
U 0D, 7700,C800,5BE4,03D4,5470,180E     0090 14526 ;% 0D    WDR_UNROT(R[LONLIT])            ; validate cache location 4
U 0E, 7700,C800,5BE4,03D8,4870,180F     009B 14530 ;% 0E    PC_R[ZERO]                      ; zero PC to force prefetch
```

```
U 0F, 7700,C800,0364,1700,0470,1810  00A6 14534 ;% 0F   BUT/IRD1TST                                  ; perform ird1
U 10, 7300,4800,0364,0300,0470,1811  00B1 14538 ;% 10   NOP,STOP.CLOCK                               ; halt
                                     00BC 14542
U 11, 7700,4800,0364,0300,0470,1812  00BC 14543 ;% 11   NOP                                         ; get cpu up to speed
U 12, 7700,4800,0364,1B00,0470,1800  00C7 14547 ;% 12   BUT/LOD.INC.BRA,NEXT/1800                   ; load OSR & increment IRD cntr
U 13, 7300,4800,0364,0300,0470,1814  00D2 14551 ;% 13   STOP.CLOCK                                  ; halt
                                     00DD 14555
U 14, 7700,C800,0364,0300,0470,1815  00DD 14556 ;% 14   NOP                                         ; get cpu up to speed
U 15, 5700,C800,0364,0300,2A70,1816  00E8 14560 ;% 15   WCTRL/FPA.ENABLE_WB5,WB_RDM                  ; write FPA enable bit
U 16, 7700,4800,0364,0300,0470,1817  00F3 14564 ;% 16   NOP                                         ; ...
U 17, 7700,4800,0364,C300,0470,1800  00FE 14568 ;% 17   BUT/NO.FPA,NEXT/1800                        ; check FPA status
U 18, 7300,C800,0364,0300,0470,1819  0109 14572 ;% 18   STOP.CLOCK                                  ; halt
                                     0114 14576
U 19, 7700,C800,0364,0300,0470,181A  0114 14577 ;% 19   NOP                                         ; get the cpu up to speed
U 1A, 7700,F800,7F9F,FFFF,8470,181B  011F 14581 ;% 1A   LONLIT_[00C00000]                            ; address of s&c bit in crar
U 1B, 7700,4800,5BE4,03D4,2470,181C  012A 14585 ;% 1B   WCTRL/LOADCRAR,WB_R[LONLIT]                 ; address the s&c bit
U 1C, 7700,B800,7FDF,FFFF,8470,181D  0135 14589 ;% 1C   LONLIT_[00400000]                            ; data for ctrl-p int. disable
U 1D, 7700,4800,5BE4,03D4,2C70,181E  0140 14593 ;% 1D   WCTRL/CONWRITE,WB_R[LONLIT]                 ; disable ctrl-p interrupts
U 1E, 7300,C800,0364,0300,0470,181F  014B 14597 ;% 1E   STOP.CLOCK                                  ; halt
                                     0156 14601
U 1F, 7700,C800,0364,0300,0470,1820  0156 14602 ;% 1F   NOP                                         ; get the cpu up to speed
U 20, 7700,F800,7F9F,FFFF,8470,1821  0161 14606 ;% 20   LONLIT_[00C00000]                            ; address of s&c bit in crar
U 21, 7700,C800,5BE4,03D4,2470,1822  016C 14610 ;% 21   WCTRL/LOADCRAR,WB_R[LONLIT]                 ; address the s&c bit
U 22, 7700,7800,7FB7,FFFF,8470,1823  0177 14614 ;% 22   LONLIT_[00900000]                            ; data for ctrl-p int. enable
U 23, 7700,4800,5BE4,03D4,2C70,1824  0182 14618 ;% 23   WCTRL/CONWRITE,WB_R[LONLIT]                 ; enable ctrl-p interrupts
U 24, 7300,C800,0364,0300,0470,1825  018D 14622 ;% 24   STOP.CLOCK                                  ; halt
                                     0198 14626
                                     0198 14627 ; Set-up RTEMP0 and RTEMP1 for error callout
                                     0198 14628
U 25, 7700,C800,0364,0300,0470,1826  0198 14629 ;% 25   NOP                                         ; get cpu up to speed
U 26, 7700,0848,5924,0300,0470,1827  01A3 14633 ;% 26   R[TEMP0]_M[TEMP8]                           ; move XB0 data to R[TEMP0]
U 27, 7700,0849,5924,0304,0470,1828  01AE 14637 ;% 27   R[TEMP1]_M[TEMP9]                           ; move XB1 data to R[TEMP1]
U 28, 7300,C800,0364,0300,0470,1829  01B9 14641 ;% 28   STOP.CLOCK                                  ; halt
                                     01C4 14645
                                     01C4 14646 RTEMP_DATA
                                     0001 14647
                         00000000    0001 14648         .LONG   0                                 ; rtemp0 = contents of xb0
                         00000000    0005 14649         .LONG   0                                 ; rtemp1 = contents of xb1
                         00000000    0009 14650         .LONG   0                                 ; rtemp2 = instruction
                                     000D 14651
                                     000D 14652 END_CPU_DATA
                                     0637 14653
                                     0637 14654 END_TEST_DATA
                                     0637
                                     0637       ;-----------------------------------------------------------------------
                                     0637 14655
                                     0637 14656 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
                0000 14658 .END
```

```
$CONTROL_C_CODE= 0000000A          CONT_FLAG       = 00000020          END_TEST_198     000005F0 R     63
$FILE_NOT_FOUND= 00000009          CONT_KEY        = 0000001C          END_TEST_199     000005DB R     69
$STN            = 00000000         CPU_RUN         = 00000010          END_TEST_200     000005F0 R     6F
$TEST_NUMBER    = 000000CE         CSAMR           = 00007404          END_TEST_201     000005DB R     75
ACV             = 0000000F         CSBUF           = 00007424          END_TEST_202     0000064C R     7B
AC_LOW          = 00000080         CSTRP           = 00007422          END_TEST_203     00000637 R     81
ADD             = 00000019         CS_ADD_BUFF_HGH= 00007425           END_TEST_204     0000064C R     87
ADDR_MATCH_HI   = 00007405         CS_ADD_BUFF_LOW= 00007424           END_TEST_205     00000637 R     8D
ADDR_MATCH_LO   = 00007404         CS_ADD_TRAP_HGH= 00007423           EN_TRACE_REG    = 00000008
ADK             = 0000000B         CS_ADD_TRAP_LOW= 00007422           ERRLOG_FLAG     = 00000000
ALK             = 00000000         CYCLE_FLAG      = 00000008          ERROR_FLAG      = 00000010
ALP             = 00000016         CYCLE_KEY       = 00000020          ERROR_LOG_ADR   = 000032FC
ARG_CNT         = 000001EE R       8D  D0          = 0000000D          ERR_LOG_INIT    = 00000001
A_REG_BYTE_0    = 00007410         D1              = 0000000C          EXP_STALL_FLAG  = 00000010
A_REG_BYTE_1    = 00007411         D2              = 00000000          EXP_UTRAP_FLAG  = 00000040
A_REG_BYTE_2    = 00007412         DATA_FLAG       = 00000000          FAC             = 00000020
A_REG_BYTE_3    = 00007413         DATA_LENGTH     = 00000004          FAULT           = 00000002
B               = 00000001         DCSAD           = 00007401          FCC             = 00000015
BADD1           = 00007426         DCSCF           = 00007403          FCS             = 00000021
BELL_FLAG       = 00000008         DCSCR           = 00007402          FETCH_FLAG      = 00000008
BELL_KEY        = 0000000E         DCSDA           = 00007400          FEX             = 0000001E
BUFF_ADDR_LOW   = 00007426         DCS_ADDR_CLEAR  = 00000002          FFH             = 00000024
BURST_STOP_FLAG= 00000080          DCS_ADDR_REG_RO= 00007427           FFL             = 00000023
BYTE            = 00000001         DCS_ADDR_REG_WO= 00007401           FIC             = 0000001F
CACHE_FLAG      = 00000000         DCS_CTRL_FILE   = 00007403          FIO             = 0000001D
CACHE_FLAG1     = 00000000         DCS_CTRL_REG    = 00007402          FLAG_KEY        = 00000004
CACHE_HEAD      = 00000000 R       48  DCS_DATA_REG= 00007400          FMR             = 00000022
CACHE_LENGTH    = 00000001         DCS_FLAG        = 00000000          FPA             = 00000002
CAK             = 0000000A         DCS_FLAG1       = 00000000          FP_BOOT_0       = 00000001
CCC             = 00000004         DCS_HEAD        = 00000000 R       8E  FP_BOOT_1    = 00000002
CCS             = 00000005         DCS_LENGTH      = 00000001          FP_START_0      = 00000004
CF_KEY          = 00000024         DCS_SCRATCH_ADR= 00000036           FP_START_1      = 00000008
CLA             = 00000003         DCS_START       = 00001800          FQA             = 00000014
CLEAR_CTRL_FILE= 00000002          DC_LOW          = 00000040          FRONT1          = 00007420
CLKDCS          = 00007427         DIAGNOSE_FLAG   = 00000004          FRONT2          = 00007421
CLK_CTRL_0      = 00000020         DPM             = 00000000          FRONT_PNL_1     = 00007420
CLK_CTRL_0_RO   = 00000040         DUM1            = FFFFFFFD          FRONT_PNL_2     = 00007421
CLK_CTRL_1      = 00000040         ENDTEST_LOC     = 0000063C R       8D  FRONT_PNL_LOCK = 00000080
CLK_CTRL_1_RO   = 00000080         END_SLICE       = 00000025          HALT_FLAG       = 00000001
CLOCK_STOPED    = 00000080         END_TEST_182     000000F2 R     02  HALT_KEY       = 00000008
CMI             = 00000007         END_TEST_183     00000077 R     09  HALT_ON_MATCH  = 00000001
CMICTL          = 0000741C         END_TEST_184     00000078 R     0F  HEAD           = FFFF7C00
CMI_COMPLETE    = 00000020         END_TEST_185     00000154 R     15  HIGH           = 00000001
CMI_CTRL_CLEAR  = 00000001         END_TEST_186     000000FC R     1B  H_FROM_WBUS    = 00000010
CMI_CTRL_REG    = 0000741C         END_TEST_187     0000009C R     21  IB_FLAG        = 00000020
CMI_GO          = 00000008         END_TEST_188     00000075 R     27  IB_KEY         = 00000012
CMI_READ        = 00000040         END_TEST_189     000000C9 R     2D  INSTR_FLAG     = 00000004
CMI_STATUS_0    = 00000008         END_TEST_190     0000006C R     33  INSTR_KEY      = 00000018
CMI_STATUS_1    = 00000010         END_TEST_191     0000006A R     39  INT            = 00000010
CMI_WRITE       = 00000020         END_TEST_192     0000009C R     3F  IRD            = 00000005
CML             = 0000000E         END_TEST_193     00000081 R     45  I_INDEX_FLAG   = 00000000
COMPOSIT_LENGTH= 00000011          END_TEST_194     0000058D R     4B  I_INIT         = 00000000
CON             = 00000011         END_TEST_195     00000578 R     51  J_INDEX_FLAG   = 00000000
CONTINUE_FLAG   = 00000002         END_TEST_196     0000058D R     57  J_INIT         = 00000000
CONTROL_C_FLAG  = 00000001         END_TEST_197     00000578 R     5D  K_INDEX_FLAG   = 00000000
```

```
K_INIT            = 00000000          MT_KEY            = 0000002E          RDM_KEY           = 0000002A
L                 = 00000004          N                 = 00000001          RDM_LAST          = 00000008
LIST_END          = 000001F4 R    8D  NER_FLAG          = 00000004          RD_CTRL_REG       = 0000741E
LIST_HEAD         = 000001EF R    8D  NER_KEY           = 0000000C          REMOTE            = 00000001
LIST_LENGTH       = 00000005          NOERROR           = 00000001          RTEMP_FLAG        = 00000000
LITERAL           = 00000000          NOTEQUAL          = 00000003          RTEMP_FLAG1       = 00000000
LONG              = 00000004          ONEQUAL           = 00000002          RTEMP_HEAD        = 00000000 R    8F
LOOP_ERROR_FLAG   = 00000020          ONERROR           = 00000000          RTEMP_LENGTH      = 00000001
LOOP_FLAG         = 00000002          OP_BEGIN_LOOP     = 00000008          RT_KEY            = 0000002C
LOOP_KEY          = 0000000A          OP_BURST_CLOCK    = 00000002          SAC               = 00000009
LOST_FLAG         = 00000010          OP_COMPARE_REG    = 00000004          SA_CLOCK          = 00000080
LOST_KEY          = 0000001A          OP_COMPARE_REGM   = 00000006          SA_FLAG           = 00000010
LOW               = 00000000          OP_COMPARE_VB     = 00000020          SA_KEY            = 00000010
M                 = 0000000A          OP_DUMPLOG        = 00000030          SA_START_STOP     = 00000080
M$TEST_SIZE       = 000007A2          OP_ENABL_STALL    = 00000038          SBE_FLAG          = 00000020
M$TEST_SIZE_D     = 000003E2          OP_ENABL_TRAP     = 0000002E          SBE_KEY           = 00000042
MA0               = 00000008          OP_END_FILE       = 00000024          SECTION_FLAG      = 00000080
MA1               = 0000000A          OP_END_LOOP       = 0000000A          SF_KEY            = 00000040
MAD               = 00000013          OP_END_TEST       = 0000000C          SOMM_FLAG         = 00000001
MAIN32            = 0000741D          OP_ERRLOG         = 00000032          SOMM_KEY          = 00000006
MAINT_A_TO_CMI    = 00000080          OP_ERROR_LOOP     = 0000000E          SPA               = 00000002
MAINT_CMI_TO_MH   = 00000020          OP_FETCH          = 00000022          SRK               = 00000001
MAINT_DCS_ENABL   = 00000004          OP_IF_ERROR       = 00000012          SRM               = 00000017
MAINT_MD_TO_CMI   = 00000040          OP_INC_INDEX      = 0000002A          SR_KEY            = 0000003C
MAINT_REG         = 0000741D          OP_INIT           = 00000014          STACK_SIZE        = 00000400
MAINT_STB_INH     = 00000002          OP_LOAD_DCS       = 00000000          STATUS            = 00007406
MAINT_TRAP_OFF    = 00000001          OP_LOAD_FIELD     = 00000026          STATUS_REG        = 00007406
MAINT_WB_TO_WH    = 00000008          OP_LOAD_INDEX     = 00000028          STEP_KEY          = 0000001E
MAINT_WD_TO_WB    = 00000010          OP_LOAD_REG       = 00000016          STOP_CLOCK        = 00000004
MAP               = 00000012          OP_MASK           = 00000018          STROBE_CMI        = 00000040
MAREG             = 00007410          OP_NEW_TEST       = 0000001A          ST_KEY            = 0000001E
MASTER_HALT_EN    = 00000008          OP_PATTERN_GEN    = 00000010          TEMP              = 00000002 R    8C
MAX_ARGUMENTS     = 00000010          OP_PRINT_N        = 00000036          TEMP1             = 0000002C
MA_KEY            = 00000038          OP_PRINT_S        = 00000034          TEMP2             = 000001D2
MC0               = 00000003          OP_SAVE_INDEX     = 0000002C          TEMP3             = 00000002
MC1               = 00000009          OP_SKIP           = 0000001C          TEMP4             = 00000002 R    8C
MDL               = 0000001B          OP_SUB_TEST       = 0000001E          TEST              = 00000004
MDR               = 00000018          OVERLAY_HEAD      = 00000000 R    92  TEST_FLAG         = 00000000
MDREG             = 00007414          PARITY_ERROR      = 00000020          TEST_HEAD         = 0000002F R    8D
MD_KEY            = 00000034          PAR_CHK_ENABL     = 00000008          TEST_KEY          = 00000000
MD_REG_BYTE_0     = 00007414          PAR_STOP_ENABL    = 00000010          TEST_LENGTH       = 0000000D
MD_REG_BYTE_1     = 00007415          PASS_KEY          = 00000002          TEST_SECT_HEAD    = 00000000 R    8D
MD_REG_BYTE_2     = 00007416          PB_INIT           = 00000040          THIS_TEST         = 000000CD
MD_REG_BYTE_3     = 00007417          PB_KEY            = 00000036          TICK_FLAG         = 00000002
MEC               = 0000001A          PC_KEY            = 00000030          TICK_KEY          = 00000022
MHREG             = 00007418          PHB               = 00000007          TOK               = 00000006
MH_REG_BYTE_0     = 00007418          PRK               = 0000000C          TRACE_ENABL       = 00000004
MH_REG_BYTE_1     = 00007419          PS_KEY            = 0000003E          TRAP_HALT_EN      = 00000020
MH_REG_BYTE_2     = 0000741A          QA_FLAG           = 00000040          TRAP_OFF          = 00000002
MH_REG_BYTE_3     = 0000741B          QA_KEY            = 00000014          TR_FLAG           = 00000080
MIC               = 00000001          QV_FLAG           = 00000002          TR_KEY            = 00000016
MICROWORD         = 0000000A          QV_KEY            = 00000028          TSTSPAN_FLAG      = 00000040
MICRO_ADDR_INH    = 00000080          RDCTRL            = 0000741E          UBI               = 00000006
MONITOR_SIZE      = 00002C5E          RDM               = 00000004          UDP               = 0000001C
MSQ               = 00000008          RDM_FLAG          = 00000004          UTR               = 0000000D
```

```
VA_KEY          = 00000032
VBUS_CLOCK      = 00000080
VBUS_COMPARE    = 00000080
VBUS_LOAD       = 00000010
VBUS_SERIAL_IN  = 00000040
VBUS_SERIAL_OUT= 00000001
VB_KEY          = 00000026
V_BUS_STROBE    = 00000001
W               = 00000002
WBUS_FROM_D     = 00000020
WDREG           = 0000742C
WD_KEY          = 0000003A
WD_REG_BYTE_0   = 0000742C
WD_REG_BYTE_1   = 0000742D
WD_REG_BYTE_2   = 0000742E
WD_REG_BYTE_3   = 0000742F
WHREG           = 00007430
WH_REG_BYTE_0   = 00007430
WH_REG_BYTE_1   = 00007431
WH_REG_BYTE_2   = 00007432
WH_REG_BYTE_3   = 00007433
WORD            = 00000002
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
X_0182_D	00000002 (2.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0182_T	00000100 (256.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0182_GDCS	00000022 (34.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0182_ERTEMP	00000001 (1.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0182_FCACHE	00000001 (1.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0182_HFILL	0000005A (90.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
DIRECTORY	00000018 (24.)	07 (7.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0183_D	00000002 (2.)	08 (8.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0183_T	00000080 (128.)	09 (9.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0183_GDCS	00000038 (56.)	0A (10.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0183_ERTEMP	00000001 (1.)	0B (11.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0183_FCACHE	00000001 (1.)	0C (12.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0183_HFILL	00000044 (68.)	0D (13.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0184_D	00000002 (2.)	0E (14.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0184_T	00000080 (128.)	0F (15.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0184_GDCS	00000038 (56.)	10 (16.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0184_ERTEMP	00000001 (1.)	11 (17.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0184_FCACHE	00000001 (1.)	12 (18.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0184_HFILL	00000044 (68.)	13 (19.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0185_D	00000002 (2.)	14 (20.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0185_T	00000180 (384.)	15 (21.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0185_GDCS	0000002D (45.)	16 (22.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0185_ERTEMP	00000001 (1.)	17 (23.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0185_FCACHE	00000001 (1.)	18 (24.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0185_HFILL	0000004F (79.)	19 (25.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0186_D	00000002 (2.)	1A (26.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0186_T	00000180 (384.)	1B (27.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0186_GDCS	0000002D (45.)	1C (28.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0186_ERTEMP	00000001 (1.)	1D (29.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0186_FCACHE	00000001 (1.)	1E (30.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0186_HFILL	0000004F (79.)	1F (31.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0187_D	00000002 (2.)	20 (32.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0187_T	00000100 (256.)	21 (33.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0187_GDCS	0000002D (45.)	22 (34.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0187_ERTEMP	00000001 (1.)	23 (35.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0187_FCACHE	00000001 (1.)	24 (36.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0187_HFILL	0000004F (79.)	25 (37.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0188_D	00000002 (2.)	26 (38.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0188_T	00000080 (128.)	27 (39.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0188_GDCS	0000002D (45.)	28 (40.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0188_ERTEMP	00000001 (1.)	29 (41.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0188_FCACHE	00000001 (1.)	2A (42.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0188_HFILL	0000004F (79.)	2B (43.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0189_D	00000002 (2.)	2C (44.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0189_T	00000100 (256.)	2D (45.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0189_GDCS	00000038 (56.)	2E (46.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0189_ERTEMP	00000009 (9.)	2F (47.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

X_0189_FCACHE	00000001	(1.)	30	(48.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0189_HFILL	0000003C	(60.)	31	(49.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0190_D	00000002	(2.)	32	(50.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0190_T	00000080	(128.)	33	(51.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0190_GDCS	00000043	(67.)	34	(52.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0190_ERTEMP	00000001	(1.)	35	(53.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0190_FCACHE	00000001	(1.)	36	(54.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0190_HFILL	00000039	(57.)	37	(55.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0191_D	00000002	(2.)	38	(56.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0191_T	00000080	(128.)	39	(57.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0191_GDCS	0000002D	(45.)	3A	(58.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0191_ERTEMP	00000001	(1.)	3B	(59.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0191_FCACHE	00000001	(1.)	3C	(60.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0191_HFILL	0000004F	(79.)	3D	(61.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0192_D	00000002	(2.)	3E	(62.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0192_T	00000100	(256.)	3F	(63.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0192_GDCS	00000038	(56.)	40	(64.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0192_ERTEMP	00000005	(5.)	41	(65.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0192_FCACHE	00000001	(1.)	42	(66.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0192_HFILL	00000040	(64.)	43	(67.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0193_D	00000002	(2.)	44	(68.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0193_T	00000100	(256.)	45	(69.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0193_GDCS	00000064	(100.)	46	(70.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0193_ERTEMP	0000000D	(13.)	47	(71.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0193_FCACHE	00000009	(9.)	48	(72.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0193_HFILL	00000004	(4.)	49	(73.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0194_D	00000002	(2.)	4A	(74.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0194_T	00000600	(1536.)	4B	(75.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0194_GDCS	00000161	(353.)	4C	(76.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0194_ERTEMP	00000009	(9.)	4D	(77.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0194_FCACHE	00000001	(1.)	4E	(78.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0194_HFILL	00000013	(19.)	4F	(79.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0195_D	00000002	(2.)	50	(80.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0195_T	00000580	(1408.)	51	(81.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0195_GDCS	00000161	(353.)	52	(82.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0195_ERTEMP	00000009	(9.)	53	(83.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0195_FCACHE	00000001	(1.)	54	(84.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0195_HFILL	00000013	(19.)	55	(85.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0196_D	00000002	(2.)	56	(86.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0196_T	00000600	(1536.)	57	(87.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0196_GDCS	00000177	(375.)	58	(88.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0196_ERTEMP	00000009	(9.)	59	(89.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0196_FCACHE	00000001	(1.)	5A	(90.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0196_HFILL	0000007D	(125.)	5B	(91.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0197_D	00000002	(2.)	5C	(92.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0197_T	00000580	(1408.)	5D	(93.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0197_GDCS	00000177	(375.)	5E	(94.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0197_ERTEMP	00000009	(9.)	5F	(95.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0197_FCACHE	00000001	(1.)	60	(96.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0197_HFILL	0000007D	(125.)	61	(97.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0198_D	00000002	(2.)	62	(98.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0198_T	00000600	(1536.)	63	(99.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0198_GDCS	000001C4	(452.)	64	(100.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0198_ERTEMP	0000000D	(13.)	65	(101.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0198_FCACHE	00000001	(1.)	66	(102.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE

X_0198_HFILL	0000002C	(44.)	67 (103.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0199_D	00000002	(2.)	68 (104.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0199_T	00000600	(1536.)	69 (105.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0199_GDCS	000001C4	(452.)	6A (106.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0199_ERTEMP	0000000D	(13.)	6B (107.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0199_FCACHE	00000001	(1.)	6C (108.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0199_HFILL	0000002C	(44.)	6D (109.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0200_D	00000002	(2.)	6E (110.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0200_T	00000600	(1536.)	6F (111.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0200_GDCS	000001C4	(452.)	70 (112.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0200_ERTEMP	0000000D	(13.)	71 (113.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0200_FCACHE	00000001	(1.)	72 (114.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0200_HFILL	0000002C	(44.)	73 (115.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0201_D	00000002	(2.)	74 (116.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0201_T	00000600	(1536.)	75 (117.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0201_GDCS	000001C4	(452.)	76 (118.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0201_ERTEMP	0000000D	(13.)	77 (119.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0201_FCACHE	00000001	(1.)	78 (120.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0201_HFILL	0000002C	(44.)	79 (121.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0202_D	00000002	(2.)	7A (122.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0202_T	00000680	(1664.)	7B (123.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0202_GDCS	000001C4	(452.)	7C (124.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0202_ERTEMP	0000000D	(13.)	7D (125.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0202_FCACHE	00000001	(1.)	7E (126.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0202_HFILL	0000002C	(44.)	7F (127.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0203_D	00000002	(2.)	80 (128.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0203_T	00000680	(1664.)	81 (129.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0203_GDCS	000001C4	(452.)	82 (130.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0203_ERTEMP	0000000D	(13.)	83 (131.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0203_FCACHE	00000001	(1.)	84 (132.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0203_HFILL	0000002C	(44.)	85 (133.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0204_D	00000002	(2.)	86 (134.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0204_T	00000680	(1664.)	87 (135.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0204_GDCS	000001C4	(452.)	88 (136.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0204_ERTEMP	0000000D	(13.)	89 (137.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0204_FCACHE	00000001	(1.)	8A (138.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0204_HFILL	0000002C	(44.)	8B (139.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0205_D	00000002	(2.)	8C (140.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0205_T	00000680	(1664.)	8D (141.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0205_GDCS	000001C4	(452.)	8E (142.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0205_ERTEMP	0000000D	(13.)	8F (143.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0205_FCACHE	00000001	(1.)	90 (144.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0205_HFILL	0000002C	(44.)	91 (145.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0206_D	00000000	(0.)	92 (146.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	41	00:00:00.10	00:00:07.82
Command processing	880	00:00:00.69	00:00:06.23
Pass 1	3653	00:01:38.31	00:03:37.90
Symbol table sort	0	00:00:00.49	00:00:00.60

Pass 2	45	00:00:30.50	00:01:20.26
Symbol table output	2	00:00:00.14	00:00:00.14
Psect synopsis output	2	00:00:00.34	00:00:00.95
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	4623	00:02:10.57	00:05:13.90

The working set limit was 6450 pages.
1473054 bytes (2878 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 352 non-local and 229 local symbols.
15400 source lines were read in Pass 1, producing 357 object records in Pass 2.
96 pages of virtual memory were used to define 44 macros.

+----------------------------+
! Macro library statistics !
+----------------------------+

Macro library name	Macros defined
DISK$UCODPACK00:[VAX750]COMETMAC.MLB;805	44
SYS$COMMON:[SYSLIB]STARLET.MLB;2	0
TOTALS (all libraries)	44

1810 GETS were required to define 44 macros.

There were 0 errors, 12 warnings and 0 information messages, on lines:
 2064 (1) 3496 (1) 3990 (1) 5434 (1) 6090 (1)
 7699 (1) 8355 (1) 9964 (1) 10660 (1) 12310 (1)
13005 (1) 14656 (1)

MACRO UCOD$0:[VAX750.DPM]TITLE+DISK$UCODPACK00:[VAX750]COMETASM+UCOD$0:[VAX750.DPM]ECKAB5/LIS=ECKAB5/OBJ=ECKAB5

(1) 24 "DPM Revision History"
(1) 1 ""
(1) 3 "DIAGNOSTIC MICROCODE MACROS"
(1) 98 " RDM Control File Macros"
(1) 109 " Diagnostic Macros"
(1) 194 ""
(1) 195 "MODULE GATE ARRAY MAPS"
(1) 197 " L0001 MULTIPLIER AND GATE ARRAY LOCATION"
(1) 254 " L0002 GATE ARRAY LOCATION"
(1) 311 " L0003 GATE ARRAY LOCATION"
(1) 369 " L0004 GATE ARRAY LOCATION"
(1) 427 " L0011 & L0016 GATE ARRAY LOCATION"
(1) 485 ""
(1) 490 "EQUATED SYMBOLS
(1) 493 ""
(1) 2 "TEST 0CE CM Odd Address BUT Test
(1) 397 "TEST 0CF CM IR2.T0.0 BUT Test
(1) 678 "TEST 0D0 CM IR2 BUT Test
(1) 960 "TEST 0D1 CM IR5 BUT Test
(1) 1345 "TEST 0D2 CM Branch on Address Test
(1) 1631 "TEST 0D3 CM OS Branch Test
(1) 1890 "TEST 0D4 COMPATABILITY MODE DSIZE ROM TEST 1
(1) 2278 "TEST 0D5 COMPATABILITY MODE DSIZE ROM TEST 2
(1) 2721 "TEST 0D6 CM IRD1 Instruction Address Decode Test 1
(1) 3103 "TEST 0D7 CM IRD1 Instruction Address Decode Test 2
(1) 3385 "TEST 0D8 CM First IRDX Instruction Address Decode Test 1
(1) 3796 "TEST 0D9 CM First IRDX Instruction Address Decode Test 2
(1) 4111 "TEST 0DA CM Second IRDX Instruction Address Decode Test 1
(1) 4561 "TEST 0DB CM Second IRDX Instruction Address Decode Test 2
(1) 4929 "TEST 0DC TEST CCS FOR PARITY ERRORS
(1) 5023 "TEST 0DD RSRC/ZERO TIMING TEST
(1) 5142 "TEST 0DE FPA ENABLE/DISABLE TEST
(1) 5302 "TEST 0DF TEST FPA STALL AND WAIT FUNCTIONS
(1) 5433 "TEST 0E0 FPA MBUS/WBUS INTERFACE TEST
(1) 5608 "TEST 0E1 TEST FPA RESERVED OPERAND TRAP
(1) 5789 "TEST 0E2 TEST FPA TRAP LOGIC
(1) 5972 "TEST 0E3 FPA CONDITION CODES TEST 1
(1) 6143 "TEST 0E4 FPA CONDITION CODES TEST 2
(1) 6332 "TEST 0E5 FPA CONDITION CODES TEST 3
(1) 6535 "TEST 0E6 FPA CONDITION CODES TEST 4
(1) 6718 "TEST 0E7 TEST FPA COPY OF CONDITION CODES
(1) 6898 "TEST 0E8 TEST FPA COPY OF FU BIT

```
0000     1          .TITLE  ECKAB VAX-11/750 DPM MICRO DIAGNOSTICS
0000     2          .IDENT  /V09.02/
0000     3 ;
0000     4 ; COPYRIGHT (C) 1980,1983
0000     5 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     6 ;
0000     7 ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE  ONLY ON A SINGLE
0000     8 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000     9 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    10 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    11 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    12 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    13 ; REMAIN IN DEC.
0000    14 ;
0000    15 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    16 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    17 ; CORPORATION.
0000    18 ;
0000    19 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    20 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    21 ;
0000    22 ;
```

```
0000     24          .SBTTL  "DPM Revision History"
0000     25 ;++
0000     26 ; DPM MICRO-DIAGNOSTIC PACKAGE
0000     27 ;
0000     28 ; REVISION HISTORY
0000     29 ;
0000     30 ; 00-01 Tom Groetzinger 4-FEB-1980
0000     31 ;       Started file.
0000     32 ;
0000     33 ; 00-02 Dave Spain      27-FEB-1980
0000     34 ;       Added CS BUS INTEGRITY test.
0000     35 ;
0000     36 ; 00-03 Tom Groetzinger 12-MAR-1980
0000     37 ;       Added five tests.
0000     38 ;
0000     39 ; 00-04 Tom Groetzinger 17-MAR-1980
0000     40 ;       Added SRK,SRM callouts to some tests.
0000     41 ;
0000     42 ; 00-05 Tom Groetzinger 17-MAR-1980
0000     43 ;       Improved tests 7 and 16 to increase coverage due to Westboro feedback
0000     44 ;
0000     45 ; 00-06 Tom Groetzinger 17-MAR-1980
0000     46 ;       Debugged test 16 dueu to major rewrite
0000     47 ;
0000     48 ; 00-07 Tom Groetzinger 17-MAR-1980
0000     49 ;       Added 9 Super Rotator tests
0000     50 ;
0000     51 ; 01-08 Dave Spain       4-APR-80
0000     52 ;       Added 4 Micro-sequencer tests for offical release.
0000     53 ;
0000     54 ; 01-09 Bill Landry     4-APR-1980
0000     55 ;       Added the XB, PREFETCH and IRD tests from ECKAC.SRC so
0000     56 ;       manufacturing can run one tape for DPM.
0000     57 ;
0000     58 ; 00-10 Bill Landry     4-APR-1980
0000     59 ;       Let's save version 1.0 for our first SDC release.
0000     60 ;
0000     61 ; 00-11 Tom Groetzinger 7-APR-1980
0000     62 ;       Added 2 ALK Tests
0000     63 ;
0000     64 ; 00-12 Tom Groetzinger 8-APR-1980
0000     65 ;       Deleted last 2 D Register outputs from ALK Test #6
0000     66 ;
0000     67 ; 00-13 Tom Groetzinger 14-APR-1980
0000     68 ;       Added D Register output to ALK test #6
0000     69 ;
0000     70 ; 00-14 Dave Spain      25-APR-1980
0000     71 ;       Changed mask on the micro-sequencer tests to ignore bit 13 of the
0000     72 ;       Control Store address.
0000     73 ;
0000     74 ; 00-15 Dave Spain      28-APR-1980
0000     75 ;       Fixed bug in ALK Tests 6 and 7 by replacing VSIZE/1 micro order with a
0000     76 ;       SIZE[LONG] macro in micro words 6 and 3 respectively.
0000     77 ;
0000     78 ; 00-16 Dave Spain      29-APR-1980
0000     79 ;       Changed mask on CS address for Instruction Address Decode test from
0000     80 ;       3FFF to 1FFF to ignore bit 13 of Control Store address.
```

```
0000     81 ;
0000     82 ; 00-17 Dave Spain      30-APR-1980
0000     83 ;       Added CCS module to error callout for Control Store Bus Integrity test.
0000     84 ;
0000     85 ; 00-18 Dave Spain       9-MAY-1980
0000     86 ;       Added SRM gate array to error callout routine in RTMP0 and RTEMP
0000     87 ;       Sratchpad tests.
0000     88 ;
0000     89 ; 00-19 Bill Landry     10-JUN-1980
0000     90 ;       Added RDM tests to beginning of tape
0000     91 ;
0000     92 ; 01-00 Bill Landry/Dave Spain  13-JUN-1980
0000     93 ;       Added macro file to title section and compatibility mode IRD tests
0000     94 ;       then submitted to RELEASE ENGINERRING.
0000     95 ;
0000     96 ; 01-01 Dave Spain      19-JUN-1980
0000     97 ;       Fixed prefetch incrementer test to clear out cache data after every
0000     98 ;       prefetch. This will prevent the test from mistaking bad addresses from
0000     99 ;       good ones.
0000    100 ;
0000    101 ; 01-02 Dave Spain       2-JUL-1980
0000    102 ;       Added CM Second IRDX tests to file.
0000    103 ;
0000    104 ; 01-03 Dave Spain      18-AUG-1980
0000    105 ;       Added CM branch test to file.
0000    106 ;
0000    107 ; 01-04 Dave Spain      10-SEPT-1980
0000    108 ;       Fixed "DEFAULT MTEMP0 TEST"
0000    109 ;
0000    110 ; 01-05 Bill Landry     11-SEPT-1980
0000    111 ;       Revised RDM to CMI tests
0000    112 ;
0000    113 ; 02-00 Dave Spain      28-SEPT-1980
0000    114 ;       Added rotator tests to file and submitted to SDC.
0000    115 ;
0000    116 ; 02-01 Dave Spain      28-OCT-1980
0000    117 ;       Added QV flag to tests to be run by manufacturing.
0000    118 ;
0000    119 ; 02-02 Bill Landry     10-NOV-1980
0000    120 ;       Added tests for TOK and SAC gate arrays.
0000    121 ;
0000    122 ; 02-03 Bill Landry     13-NOV-1980
0000    123 ;       Added test of REI CHECK
0000    124 ;
0000    125 ; 03-00 Dave Spain      17-NOV-1980
0000    126 ;       Added rotator tests 23-28 and released to SDC.
0000    127 ;
0000    128 ; 03-01 Dave Spain      10-DEC-1980
0000    129 ;       Added rotator tests 29-42.
0000    130 ;
0000    131 ; 03-02 Bill Landry     11-DEC-1980
0000    132 ;       Fixed bug in CS ADDRESS test
0000    133 ;
0000    134 ; 03-03 Dave Spain      12-DEC-1980
0000    135 ;       Enhanced Rotator Test 1 to call out ALP slices.
0000    136 ;
0000    137 ; 03-04 Bill Landry     16-DEC-1980
```

```
0000    138 ;       Added DSIZE ROM tests
0000    139 ;
0000    140 ; 03-05 Bill Landry     31-DEC-1980
0000    141 ;       Last great act of 80. Separated DPM and RDM tests into two files.
0000    142 ;
0000    143 ; 04-00 Bill Landry     5-JAN-1981
0000    144 ;       Submitted to RELEASE ENGINERRING
0000    145 ;
0000    146 ; 04-01 Bill Landry     14-JAN-1981
0000    147 ;       Revised INTERVAL TIMER INTERRUPT TEST to fix bug caused by INT PEND
0000    148 ;       ECO
0000    149 ;
0000    150 ; 04-02 Dave Spain      16-JAN-1981
0000    151 ;       Fixed error callout in CS Parity Error, XB, and Prefetch Incrementer
0000    152 ;       tests and added seven new rotator tests.
0000    153 ;
0000    154 ; 04-03 Bill Landry     4-FEB-1981
0000    155 ;       Added tests for CCC gate array
0000    156 ;
0000    157 ; 04-04 Dave Spain      9-FEB-1981
0000    158 ;       Changed the names of rotator tests 23-33.
0000    159 ;
0000    160 ; 04-05 Bill Landry     24-FEB-1981
0000    161 ;       Added tests for DPM discrete branching logic.
0000    162 ;
0000    163 ; 04-06 Bill Landry     27-FEB-1981
0000    164 ;       Added tests for branching logic in PHB gate array.
0000    165 ;
0000    166 ; 05-00 Dave Spain       1-MAR-1981
0000    167 ;       Added 5 SRK status rotator tests and submitted to SDC.
0000    168 ;
0000    169 ; 05-01 Dave Spain      25-MAR-1981
0000    170 ;       Fixed a loop on test bug in the Alu Functions Test by adding a LDFIELD
0000    171 ;       pseudo-op that reinitializes the ROT field in Subtest 1.
0000    172 ;
0000    173 ; 05-02 Bill Landry     25-MAR-1981
0000    174 ;       Fixed bug in BUT/NO.FPA test.  I was branching on the wrong conditions.
0000    175 ;
0000    176 ; 05-03 Dave Spain      14-APR-1981
0000    177 ;       Cleaned up documentation for R Scratchpads Write All Cells Test.
0000    178 ;
0000    179 ; 06-00 Bill Landry     2-MAY-1981
0000    180 ;       Added SPA branching, CON registers, TU58 registers, and CCS PARITY
0000    181 ;       tests.  Submitted to Release Engineering.
0000    182 ;
0000    183 ; 06-01 Dave Spain      17-JUN-1981
0000    184 ;       Replaced the old IRD1 Instruction Address Decode test with two new
0000    185 ;       tests that toggle the PSL's FPD bit and can handle an FPA.
0000    186 ;
0000    187 ; 06-02 Bill Landry     19-AUG-1981
0000    188 ;       Fixed bugs created by introduction of FPA option.
0000    189 ;       Replaced all references to COMET with the word CPU.
0000    190 ;       Fixed typing error in vbus data of CPU CLOCK TEST.
0000    191 ;       Submitted to RELEASE ENGINEERING.
0000    192 ;
0000    193 ; 06-03 Bill Landry     24-SEPT-1981
0000    194 ;       Fixed potential clock problem with FPA in Z BIT TEST 1.
```

```
0000    195 ;
0000    196 ; 06-04 Bill Landry     9-OCT-1981
0000    197 ;       Added changes for ECO L0002-TW002
0000    198 ;
0000    199 ; 07-00 Bill Landry     31-OCT-1981
0000    200 ;       Removed changes for ECO L0002-TW002.  ECO is on hold.
0000    201 ;       Added tests for FPA to CPU interface logic
0000    202 ;       Moved WCS test to MIC tape
0000    203 ;       Submitted to RELEASE ENGINEERING.
0000    204 ;
0000    205 ; 07-01 Bill Landry     13-NOV-1981
0000    206 ;       Stopped release of 7.0
0000    207 ;       Added FPA TRAP LOGIC TEST
0000    208 ;       Enhanced error messages for FPA
0000    209 ;       Resubmitted to RELEASE ENGINEERING
0000    210 ;
0000    211 ; 07-02 Bill Landry     20-NOV-1981
0000    212 ;       Stopped release of 7.1
0000    213 ;       Enhanced FPA ENABLE/DISABLE TEST to check for MSEQ INIT
0000    214 ;       Resubmitted to RELEASE ENGINEERING
0000    215 ;
0000    216 ; 07-03 Dave Shull      26-Mar-1982
0000    217 ;       Added the ERRLOOP pseudo to the Rotator Extract tests which were
0000    218 ;       missing these and couldn't be looped on.
0000    219 ;
0000    220 ; 07-04 Dave Shull      05-Apr-1982
0000    221 ;       Removed all BEGINSA and ENDSA pseudo's.
0000    222 ;       Submitted to RELEASE ENGINEERING
0000    223 ;
0000    224 ; 07-05 Dave Shull      21-May-1982
0000    225 ;       Add a subtest to the Micro Sequencer Stack Test that will test bits
0000    226 ;       11 and 12 of the micro stack.
0000    227 ;
0000    228 ; 07-06 Dave Shull      24-May-1982
0000    229 ;       Changed documentation in the CM Odd Address BUT Test to reflect the
0000    230 ;       ECO L0002-TW002
0000    231 ;
0000    232 ; 08-00 Dave Shull      18-Jun-1982
0000    233 ;       Added 10 new tests.  Two of those tests are an addition to the existing
0000    234 ;       IRD1 rom tests.  These 2 new tests check the signal ROM OS INH H.
0000    235 ;       The other 8 new tests check the validity of the IRDX ROMs.  They check
0000    236 ;       both the address and the signal ROM OS INH H.
0000    237 ;
0000    238 ; 08-01 Dave Shull      22-Jun-1982
0000    239 ;       Added a subtest to the Register Backup Stack test that will check the
0000    240 ;       bit that records Increment/Decrement of the GPR.
0000    241 ;
0000    242 ; 09-00 Dave Shull      03-Nov-1982
0000    243 ;       Added two new Tests for coverage the new PCS (L0008) module.
0000    244 ;
0000    245 ; 09-01 Dave Smith      15-May-1983
0000    246 ;       Correct DCS in ALK test 3,6,7. Disabled ALP's on the WBUS.
0000    247 ;
0000    248 ; 09-02 Joe Zagarella   20-Jan-1987
0000    249 ;       Fixed bug introduced by L0003-YA module in tests CE-D3, and D6-DB
0000    250 ;--
0000      1         .SBTTL  ""
```

```
0000     3          .SBTTL  "DIAGNOSTIC MICROCODE MACROS"
0000     4 ;++
0000     5 ; Revision History
0000     6 ;
0000     7 ; 28    Dave Spain      22-Jun-82
0000     8 ;       Removed LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
0000     9 ;       Decided not to use them.
0000    10 ;
0000    11 ; 27    Dave Spain      15-Jun-82
0000    12 ;       Added LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
0000    13 ;
0000    14 ; 26    Dave Spain      19-May-82
0000    15 ;       Added M[]_FPA macro.
0000    16 ;
0000    17 ; 25    Rich Lewis      23-Feb-82
0000    18 ;       Added PSL_WB, M[]_R[].OR.NIB0(M), WB_FPA CCR, WB_FPA TCR, Q_R[].ASL.P,
0000    19 ;       RDM_M[].XZ.OR.R[], FPA_WB macros
0000    20 ;
0000    21 ; 24    Dave Spain      02-Feb-82
0000    22 ;       Added FPA_R[]+M[] macro.
0000    23 ;
0000    24 ; 23    Bill Landry     22-OCT-81
0000    25 ;       Added PSL_RDM macro
0000    26 ;
0000    27 ; 22    Dave Spain       5 Oct 81
0000    28 ;       Added RDM_FPA and FPA_M[] FPA_RDM macros.
0000    29 ;
0000    30 ; 21    Bill Landry     29 Sept 81
0000    31 ;       Added FPA macro, FPA_M[] FPA_R[].
0000    32 ;
0000    33 ; 20    Dave Spain      14 Aug 81
0000    34 ;       Added the TESTFPA [] macro.
0000    35 ;
0000    36 ; 19    Dave Spain      18 July 81
0000    37 ;       Added the following FPA macros, FPA_RDM and FPA_R[].
0000    38 ;
0000    39 ; 18    Dave Spain      24 Sept 80
0000    40 ;       Added RDM_Q macro. This macro uses the ALU.
0000    41 ;
0000    42 ; 17    Bill Landry     23 Sept 80
0000    43 ;       Added macro VA_VA-ZLIT0[] for testing memory in decending order.
0000    44 ;
0000    45 ; 16    Dave Spain      27 August 80
0000    46 ;       Changed RDM_Q macro to RDM_Q Q_D. This more accurately reflects macro.
0000    47 ;
0000    48 ; 15    Bill Landry     4 June 80
0000    49 ;       Deleted BUS/PRB.RD.PTE from macro CLRTB.VA_VA and CLRCH.VA_VA to
0000    50 ;       satisfy validity checks.
0000    51 ;
0000    52 ; 14    Dave Spain      23 May 80
0000    53 ;       Added BUS/PRB.RD.PTE to the clear cache and clear TB macros for Mr.
0000    54 ;       Bill.
0000    55 ;
0000    56 ; 13    Dave Spain      14 Apr 80
0000    57 ;       Added SIZE[] or VSIZE/1 to some macros. This allows version 65 of
0000    58 ;       the COMET micro-code defines to run.
0000    59 ;
```

```
0000     60 ; 12    Bill Landry      1 Apr 80
0000     61 ;       Added BUS/PRB.RD micro order to RDM_TB macro to compensate for timing
0000     62 ;       problem in ADD gate array.  This is not an APRIL FOOL!
0000     63 ;
0000     64 ; 11    Dave Spain      27 Mar 80
0000     65 ;       Fixed TB macro's to include MSRC/VA to prevent validity failure.
0000     66 ;
0000     67 ; 10    Bill Landry     19 Mar 80
0000     68 ;       Added Q_R[].RL.P and VA_Q.OR.R[]
0000     69 ;
0000     70 ; 09    Bill Landry     19 Mar 80
0000     71 ;       Deleted TB_R[].RL.P.OR.D
0000     72 ;
0000     73 ; 08    Bill Landry     18 Mar 80
0000     74 ;       Added VA_R[].RL.P
0000     75 ;
0000     76 ; 07    Bill Landry     07 Mar 80
0000     77 ;       Added Q_M[].RL.PTE
0000     78 ;
0000     79 ; 06    Dave Spain      06 Mar 80
0000     80 ;       Changed TB_R[].RL.P macro to TB_R[].RL.P.OR.D.
0000     81 ;
0000     82 ; 05    Dave Spain      03 Mar 80
0000     83 ;       Added TB_R[].RL.P macro for Bill Landry.
0000     84 ;
0000     85 ; 04    Dave Spain      22 Feb 80
0000     86 ;       Added CLRTB.VA_R[] macro for Bill Landry.
0000     87 ;
0000     88 ; 03    Dave Spain      30 Jan 80
0000     89 ;       Added Tom Groetzinger's macro definitions.
0000     90 ;
0000     91 ; 02    Dave Spain      30 Jan 80
0000     92 ;       Added Tony Vezza's macro definitions.
0000     93 ;
0000     94 ; 01    Dave Spain      29 Jan 80
0000     95 ;       Macros initiated.
0000     96 ;--
0000     97
0000     98         .SBTTL  "       RDM Control File Macros"
0000     99
0000    100 ;STROBE.V.BUS           "RDMCF0/V.STROBE"
0000    101 ;DCS.ADDR_0             "RDMCF1/DCS.ADDR.CLR"
0000    102 ;STOP.CLOCK             "RDMCF2/STOP.CPU.CLOCK"
0000    103 ;LATCH.CS.ADDR          "RDMCF3/DIS.STROBE.CS.A"
0000    104 ;RDM_WB                 "RDMCF4/H.REG.FROM.WB"
0000    105 ;WB_RDM                 "RDMCF5/WB.FROM.D.REG"
0000    106 ;RDM_CMI                        "RDMCF6/STROBE.CMI"
0000    107 ;INH.CLOCK.SA           "RDMCF7/INH.SA.CLOCK"
0000    108
0000    109         .SBTTL  "       Diagnostic Macros"
0000    110
0000    111 ;CLOCK.EXT                      "CLKX/XTND"
0000    112 ;CLRCH.VA_RDM                   "WB_RDM,WCTRL/CLRCH.VA_WB,BUS/PRB.RD.PTE"
0000    113 ;CLRCH.VA_VA                    "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRCH.VA_WB
0000    114 ;CLRTB.VA_D                     "RSRC/ZERO,MUX/D.R1,ALU/OR,WCTRL/CLRTB.VA_WB,
0000    115 ;                                       BUS/PRB.RD.PTE"
0000    116 ;CLRTB.VA_RDM                   "WB_RDM,WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
```

```
0000   117 ;CLRTB.VA_R[]                "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/CLRTB.VA_WB,
0000   118 ;                                BUS/PRB.RD.PTE"
0000   119 ;CLRTB.VA_VA                 "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRTB.VA_WB
0000   120 ;CLRTB.VA_WB                 "WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
0000   121 ;D_LONLIT                    "RSRC/LONLIT,ALPCTL/WX_D_R.Q_D"
0000   122 ;D_M[]_LONLIT                "D_LONLIT,MSRC/a1"
0000   123 ;D_VA_0                      "RSRC/ZERO,ROT/ZERO,MUX/R.S,ALU/OR,DQ1/D_WX,
0000   124 ;                                WCTRL/VA_WB"
0000   125 ;FPA_M[] FPA_RDM                "MSRC/a1,WB_RDM,FPA/FPA_MBUS.FPA_WBUS"
0000   126 ;FPA_M[] FPA_R[]                "FPA/FPA_MBUS.FPA_WBUS,MSRC/a1,RSRC/a2,MUX/R
0000   127 ;                                ROT/ZERO,ALU/OR"
0000   128 ;FPA_RDM                       "WB_RDM,FPA/FPA_DATA.WBUS"
0000   129 ;FPA_R[]                      "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,FPA/FPA_DAT
0000   130 ;FPA_R[]+M[]                  "RSRC/a1,MSRC/a2,MUX/M.R1,ALU/A+B+CI,ALUCI/ZERO,FPA/
0000   131 ;FPA_WB                       "FPA/FPA_DATA.WBUS"
0000   132 ;MEMSCAR_R[]                  "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/MEMSCAR_WB"
0000   133 ;M[]_FPA                      "MSRC/a1,SPW/MLONG,FPA/WBUS_FPA"
0000   134 ;M[]_LONLIT                   "MSRC/a1,SPW/MLONG,ALPCTL/WX_R.Q_D,RSRC/LONLIT"
0000   135 ;M[]_R[].OR.NIB0(M)           "MSRC/a1,RSRC/a2,SPW/MLONG,ROT/GETNIB,ALU/OR,
0000   136 ;                                MUX/R.S,DQ1/NOP"
0000   137 ;M[]_RDM                      "WB_RDM,SPW/MLONG,MSRC/a1"
0000   138 ;PC_PC+RDM                    "WCTRL/PC_PC+WB,WB_RDM"
0000   139 ;PC_PC+4 MB_XB                "MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,SIZE[LONG]"
0000   140 ;PC_PC+4 RDM_XB               "RSRC/ZERO,MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,
0000   141 ;                                MUX/M.R1,ALU/OR,RDM_WB,SIZE[LONG]"
0000   142 ;PC_RDM                       "WB_RDM,WCTRL/PC_WB"
0000   143 ;PSL_RDM                      "CCPSL/PSL_WB.CCBR_ALUS,WB_RDM"
0000   144 ;PSL_WB                       "CCPSL/PSL_WB.CCBR_ALUS"
0000   145 ;Q_D WX_R                     "ALPCTL/WX_R.Q_D"
0000   146 ;Q_LONLIT                     "RSRC/LONLIT,ALPCTL/WX_S.Q_R"
0000   147 ;Q_M[].RL.PTE                 "ALPCTL/WX_Q_S,MSRC/a1,ROT/RL.MM.PTE"
0000   148 ;Q_R[].AND.Q                  "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/AND"
0000   149 ;Q_R[].ASL.P                  "ALPCTL/WX_Q_S,RSRC/a1,ROT/ASL.R.P"
0000   150 ;Q_R[].OR.Q                   "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/OR"
0000   151 ;Q_R[].RL.P                   "ALPCTL/WX_Q_S,RSRC/a1,ROT/RL.RR.P"
0000   152 ;RDM_ALUF                     "ALPCTL/WB_ALUF,RDM_WB"
0000   153 ;RDM_FPA                      "FPA/WBUS_FPA,RDM_WB"
0000   154 ;RDM_LONLIT                   "WB_LONLIT,RDM_WB"
0000   155 ;RDM_LONLIT.Q_M               "RSRC/LONLIT,ALPCTL/WX_R.Q_M,RDM_WB"
0000   156 ;RDM_M[]                      "MSRC/a1,ROT/ZERO,MUX/M.S,ALU/OR,RDM_WB"
0000   157 ;RDM_M[].OR.R[]               "WB_M[a1].OR.R[a2],RDM_WB"
0000   158 ;RDM_M[]-R[]                  "WB_M[a1]-R[a2],RDM_WB"
0000   159 ;RDM_M[].XZ.OR.R[]           "ALU/A+B+CI,MUX/R.S,ROT/XZ.MM,MSRC/a1,RSRC/a2,
               RDM_WB"
0000   160 ;
0000   161 ;RDM_PC                      "MSRC/PC,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
0000   162 ;RDM_PCBACK                  "RSRC/ZERO,MSRC/PCBACK,MUX/M.R1,ALU/OR,RDM_WB"
0000   163 ;RDM_Q                       "RSRC/ZERO,MUX/R.Q,ALU/OR,RDM_WB"
0000   164 ;RDM_Q Q_D                   "ALPCTL/WX_Q.Q_D,RDM_WB"
0000   165 ;RDM_R[]                     "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,RDM_WB"
0000   166 ;RDM_TB                      "WB_M[TB],RDM_WB,BUS/PRB.RD,SIZE[LONG]"
0000   167 ;RDM_VA                      "MSRC/VA,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
0000   168 ;RNUM_LONLIT                 "ALPCTL/WX_D_R.Q_M,MSRC/RNUM_WBUS,RSRC/LONLIT"
0000   169 ;R[]_RDM                     "WB_RDM,SPW/RLONG,RSRC/a1"
0000   170 ;R[].DT_Q                    "RSRC/a1,D_Q Q_D,SPW/RSIZE,VSIZE/1"
0000   171 ;R[]_VA                      "RSRC/a1,MSRC/VA,ROT/ZERO,MUX/M.S,ALU/OR,SPW/RLONG"
0000   172 ;R[]_WB                      "RSRC/a1,SPW/RLONG"
0000   173 ;TB_RDM                      "WCTRL/TB_WB,WB_RDM,MSRC/VA"
```

```
0000   174 ;TB_WB                          "WCTRL/TB_WB,MSRC/VA"
0000   175 ;TESTFPA []                     "TESTFPA/a1"
0000   176 ;VA_PC+LONLIT+I.PC_PC+I         "RSRC/LONLIT,MSRC/XB.PC_PC+I,ROT/ZERO,MUX/R.S,ALU/OR
0000   177 ;                                       ISTRM/ISIZE_DSIZE,WCTRL/VA_PC+I+W.PC_PC+I,
0000   178 ;                                       SIZE[LONG]"
0000   179 ;VA_Q.OR.R[]                     "WCTRL/VA_WB,RSRC/a1,ALU/OR,MUX/R.Q"
0000   180 ;VA_RDM                         "WB_RDM,WCTRL/VA_WB"
0000   181 ;VA_R[].RL.P                    "ALPCTL/WX_S,ROT/RL.RR.P,RSRC/a1,WCTRL/VA_WB"
0000   182 ;VA_VA-ZLIT0[]                  "WCTRL/VA_WB,LIT/LITRL,LITRL/a1,ROT/ZLIT0,MSRC/VA,
0000   183 ;                                       MUX/M.S,ALU/A-B-CI"
0000   184 ;VA_WB                          "WCTRL/VA_WB"
0000   185 ;WB_FPA CCR                     "FPA/WBUS_FPA.CC"
0000   186 ;WB_FPA TCR                     "WCTRL/FPTCR"
0000   187 ;WB_LONLIT                      "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
0000   188 ;WB_LONLIT.Q_M                  "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
0000   189 ;WB_M[]+R[]                     "ALU/A+B+CI,ALUCI/ZERO,MUX/M.R1,MSRC/a1,RSRC/a2"
0000   190 ;WB_R[].Q_D                     "RSRC/a1,ALPCTL/WX_R.Q_D"
0000   191 ;WDR_R[]                        "WCTRL/WDR_WB,ALU/OR,MUX/R.S,RSRC/a1,ROT/ZER
0000   192 ;WRITE.LONG R[]                 "BUS/WRITE.LNG,WCTRL/WDR_WB.UR,RSRC/a1,ROT/ZERO,
0000   193 ;                                       MUX/R.S,ALU/OR,SIZE[LONG]"
0000   194         .SBTTL  ""
0000   195         .SBTTL  "MODULE GATE ARRAY MAPS"
```

```
                        0000   197          .SBTTL  "       L0001 MULTIPLIER AND GATE ARRAY LOCATION"
                        0000   198 ;+-----------------------------------------------------------------------+
                        0000   199 ;|  +---------------+ +---------------+ +---------------+ +---------------+ |
                        0000   200 ;|  |               | |               | |               | |               | |
                        0000   201 ;|  |      FCC      | |      FQA      | |     FEX 0     | |     FEX 1     | |
                        0000   202 ;|  |               | |               | |               | |               | |
                        0000   203 ;|  +---------------+ +---------------+ +---------------+ +---------------+ |
                        0000   204 ;|                                                                       |
                        0000   205 ;|                                                                       |
                        0000   206 ;|                                                                       |
                        0000   207 ;|                                                                       |
                        0000   208 ;|                                                                       |
                        0000   209 ;|                   +---------------+ +---------------+ +---------------+ +---+
                        0000   210 ;|                   |               | |               | |               | |   |
                        0000   211 ;|                   |     FFA 5     | |     FFA 0     | |     FIO 1     | +---+
                        0000   212 ;|                   |               | |               | |               |     |
                        0000   213 ;|                   +---------------+ +---------------+ +---------------+     |
                        0000   214 ;|  +---------------+ +---------------+ +---------------+ +---------------+     |
                        0000   215 ;|  |               | |               | |               | |               |     |
                        0000   216 ;|  |     MULT 4    | |     FFA 4     | |     FCS 1     | |     FIO 0     |     |
                        0000   217 ;|  |               | |               | |               | |               |     |
                        0000   218 ;|  +---------------+ +---------------+ +---------------+ +---------------+     |
                        0000   219 ;|  +---------------+ +---------------+ +---------------+ +---------------+     |
                        0000   220 ;|  |               | |               | |               | |               |     |
                        0000   221 ;|  |     MULT 1    | |     ALU CLA   | |     FCS 2     | |     FMR 1     |     |
                        0000   222 ;|  |               | |               | |               | |               |     |
                        0000   223 ;|  +---------------+ +---------------+ +---------------+ +---------------+     |
                        0000   224 ;|  +---------------+ +---------------+ +---------------+ +---------------+     |
                        0000   225 ;|  |               | |               | |               | |               |     |
                        0000   226 ;|  |     MULT 5    | |     FFA 1     | |     FCS 0     | |     FMR 0     |     |
                        0000   227 ;|  |               | |               | |               | |               |     |
                        0000   228 ;|  +---------------+ +---------------+ +---------------+ +---------------+ +---+
                        0000   229 ;|  +---------------+ +---------------+ +---------------+ +---------------+ |   |
                        0000   230 ;|  |               | |               | |               | |               | +---+
                        0000   231 ;|  |     MULT 2    | |     FFA 2     | |     FCS 3     | |     FIO 2     |     |
                        0000   232 ;|  |               | |               | |               | |               |     |
                        0000   233 ;|  +---------------+ +---------------+ +---------------+ +---------------+     |
                        0000   234 ;|  +---------------+ +---------------+ +---------------+ +---------------+     |
                        0000   235 ;|  |               | |               | |               | |               |     |
                        0000   236 ;|  |     MULT 6    | |     FFA 6     | |     FIO 3     | |     FIO 5     |     |
                        0000   237 ;|  |               | |               | |               | |               |     |
                        0000   238 ;|  +---------------+ +---------------+ +---------------+ +---------------+     |
                        0000   239 ;|  +---------------+ +---------------+ +---------------+ +---------------+     |
                        0000   240 ;|  |               | |               | |               | |               |     |
                        0000   241 ;|  |     MULT 3    | |     FFA 3     | |     INC CLA   | |     FIO 4     |     |
                        0000   242 ;|  |               | |               | |               | |               |     |
                        0000   243 ;|  +---------------+ +---------------+ +---------------+ +---------------+     |
                        0000   244 ;|  +---------------+ +---------------+ +---------------+ +---------------+     |
                        0000   245 ;|  |               | |               | |               | |               |     |
                        0000   246 ;|  |     MULT 7    | |     FFA 7     | |     FIO 7     | |     FIO 6     |     |
                        0000   247 ;|  |               | |               | |               | |               |     |
                        0000   248 ;|  +---------------+ +---------------+ +---------------+ +---------------+     |
                        0000   249 ;|                                                                       |
                        0000   250 ;|                                                                       |
                        0000   251 ;|                                                                       |
                        0000   252 ;+-----------------------------------------------------------------------+
```

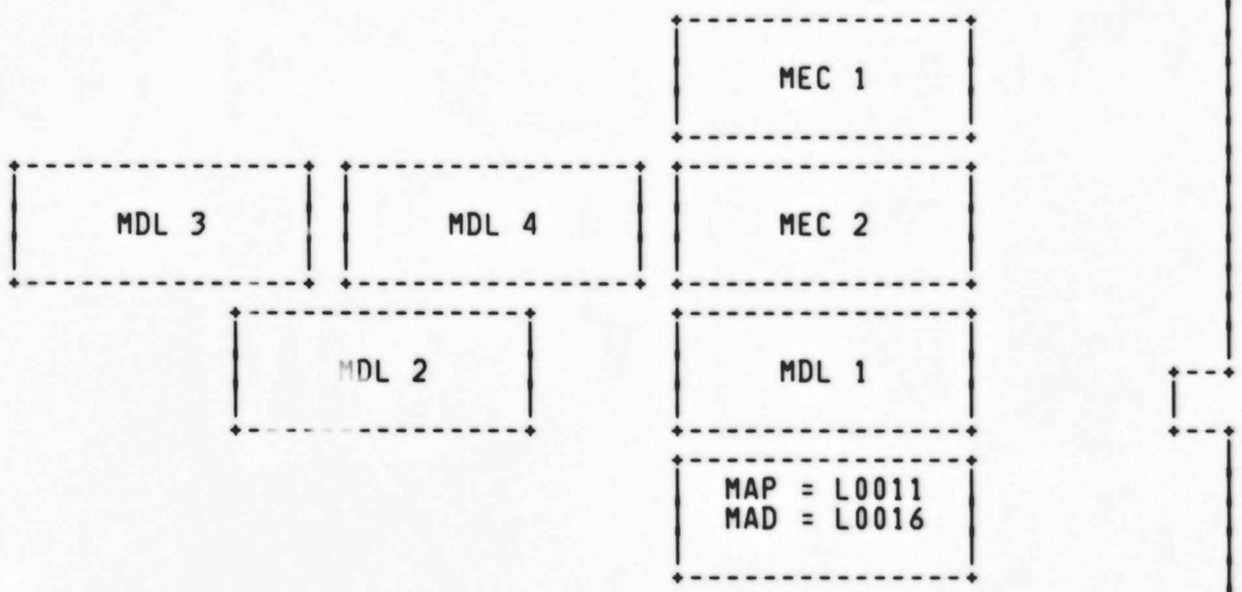
```
0000          .SBTTL  "      L0002 GATE ARRAY LOCATION"
0000   ;+-----------------------------------------------------------------------------+
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                                                                             |
0000   ;|                    +---------------+ +---------------+                      |
0000   ;|                    |               | |               |                      |
0000   ;|                    |      PHB      | |      MSQ      |                      |
0000   ;|                    |               | |               |                      |
0000   ;|                    +---------------+ +---------------+                      |
0000   ;|                    +---------------+ +---------------+                  +---+
0000   ;|                    |               | |               |                  |   |
0000   ;|                    |      CCC      | |      SAC      |                  +---+
0000   ;|                    |               | |               |                      |
0000   ;|                    +---------------+ +---------------+                      |
0000   ;|   +---------------+ +---------------+ +---------------+                      |
0000   ;|   |               | |               | |               |                      |
0000   ;|   |      SPA      | |      SRK      | |      IRD      |                      |
0000   ;|   |               | |               | |               |                      |
0000   ;|   +---------------+ +---------------+ +---------------+                      |
0000   ;|   +---------------+ +---------------+ +---------------+                      |
0000   ;|   |               | |               | |               |                      |
0000   ;|   |      TOK      | |      CLA      | |      ALK      |                      |
0000   ;|   |               | |               | |               |                      |
0000   ;|   +---------------+ +---------------+ +---------------+                      |
0000   ;|                    +---------------+ +---------------+                      |
0000   ;|                    |               | |               |                      |
0000   ;|                    |      ALP 1    | |      ALP 5    |                      |
0000   ;|                    |               | |               |                      |
0000   ;|                    +---------------+ +---------------+                  +---+
0000   ;|                    +---------------+ +---------------+                  |   |
0000   ;|                    |               | |               |                  +---+
0000   ;|                    |      ALP 2    | |      ALP 6    |                      |
0000   ;|                    |               | |               |                      |
0000   ;|                    +---------------+ +---------------+                      |
0000   ;|                    +---------------+ +---------------+                      |
0000   ;|                    |               | |               |                      |
0000   ;|                    |      ALP 3    | |      ALP 7    |                      |
0000   ;|                    |               | |               |                      |
0000   ;|                    +---------------+ +---------------+                      |
0000   ;|                    +---------------+ +---------------+                      |
0000   ;|                    |               | |               |                      |
0000   ;|                    |      ALP 4    | |      ALP 8    |                      |
0000   ;|                    |               | |               |                      |
0000   ;|                    +---------------+ +---------------+                      |
0000   ;|   +---------------+ +---------------+ +---------------+ +---------------+    |
0000   ;|   |               | |               | |               | |               |    |
0000   ;|   |      SRM 1    | |      SRM 2    | |      SRM 3    | |      SRM 4    |    |
0000   ;|   |               | |               | |               | |               |    |
0000   ;|   +---------------+ +---------------+ +---------------+ +---------------+    |
0000   ;|+-----------------------------------------------------------------------------+
```

```
                        311          .SBTTL  "       L0003 GATE ARRAY LOCATION"
                        312 ;+-----------------------------------------------------------------------+
                        313 ;|                                                                       |
                        314 ;|                                                                       |
                        315 ;|                                                                       |
                        316 ;|                                                                       |
                        317 ;|                                                                       |
                        318 ;|                                                                       |
                        319 ;|                                                                       |
                        320 ;|                                                                       |
                        321 ;|                                                                       |
                        322 ;|                                                                       |
                        323 ;|                                                                       |
                        324 ;|                                                                       |
                        325 ;|                                                                       |
                        326 ;|                                                                       |
                        327 ;|                                                                       |
                        328 ;|                                                                       |
                        329 ;|                                                                       |
                        330 ;|                                                                       |
                        331 ;|                                                                       |
                        332 ;|                                                                       |
                        333 ;|                                                                       |
                        334 ;|                                                                       |
                        335 ;|                                                                       |
                        336 ;|                                                                       |
                        337 ;|                                                                       |
                        338 ;|                                                                       |
                        339 ;|                                                                       |
                        340 ;|                                                                       |
                        341 ;|                                                                       |
                        342 ;|                                                                       |
                        343 ;|                                                                       |
                        344 ;|                                                                       |
                        345 ;|                                                                       |
                        346 ;|                                                                       |
                        347 ;|                                                                       |
                        348 ;|                                                                       |
                        349 ;|                                                                       |
                        350 ;|                                                                       |
                        351 ;|                                                                       |
                        352 ;|                                                                       |
                        353 ;|                                                                       |
                        354 ;|                                                                       |
                        355 ;|                                                                       |
                        356 ;|                                                                       |
                        357 ;|                                                                       |
                        358 ;|                                                                       |
                        359 ;|                                                                       |
                        360 ;|                                                                       |
                        361 ;|                                                                       |
                        362 ;|                                                                       |
                        363 ;|                                                                       |
                        364 ;|                                                                       |
                        365 ;|                                                                       |
                        366 ;|                                                                       |
                        367 ;+-----------------------------------------------------------------------+
```

			CAK	CMK
			ADK	UTR
			PRK	ACV
ADD 1	MDR 6	MDR 1		
ADD 2	MDR 8	MDR 4		
ADD 3	MDR 5	MDR 2		
ADD 4	MDR 7	MDR 3		

```
                        0000   369          .SBTTL  "       L0004 GATE ARRAY LOCATION"
                        0000   370 ;+-----------------------------------------------------------------------+
                        0000   371 ;|                                                                       |
                        0000   372 ;|                                                                       |
                        0000   373 ;|                                                                       |
                        0000   374 ;|                                                                       |
                        0000   375 ;|                                                                       |
                        0000   376 ;|                                                                       |
                        0000   377 ;|                                                                       |
                        0000   378 ;|                                                                       |
                        0000   379 ;|                                                                       |
                        0000   380 ;|                                                                       |
                        0000   381 ;|                    +---------------+                                  |
                        0000   382 ;|                    |               |                                  |
                        0000   383 ;|                    |      UCN      |                                  |
                        0000   384 ;|                    |               |                                  |
                        0000   385 ;|                    +---------------+                                  |
                        0000   386 ;|                    +---------------+                                  |
                        0000   387 ;|                    |               |                                  |
                        0000   388 ;|                    |     UDP 1     |                                  |
                        0000   389 ;|                    |               |                                  |
                        0000   390 ;|                    +---------------+                                  |
                        0000   391 ;|                    +---------------+                                  |
                        0000   392 ;|                    |               |                                  |
                        0000   393 ;|                    |     UDP 2     |                                  |
                        0000   394 ;|                    |               |                                  |
                        0000   395 ;|                    +---------------+                                  |
                        0000   396 ;|                    +---------------+                                  |
                        0000   397 ;|                    |               |                                  |
                        0000   398 ;|                    |     UDP 4     |                                  |
                        0000   399 ;|                    |               |                                  |
                        0000   400 ;|                    +---------------+                                  |
                        0000   401 ;|                    +---------------+                                  |
                        0000   402 ;|                    |               |                                  |
                        0000   403 ;|                    |     UDP 3     |                                  |
                        0000   404 ;|                    |               |                                  |
                        0000   405 ;|                    +---------------+                                  |
                        0000   406 ;|                    +---------------+                                  |
                        0000   407 ;|                    |               |                                  |
                        0000   408 ;|                    |      INT      |                                  |
                        0000   409 ;|                    |               |                                  |
                        0000   410 ;|                    +---------------+                                  |
                        0000   411 ;|                    +---------------+                                  |
                        0000   412 ;|                    |               |                                  |
                        0000   413 ;|                    |     CON 1     |                                  |
                        0000   414 ;|                    |               |                                  |
                        0000   415 ;|                    +---------------+                                  |
                        0000   416 ;|                    +---------------+                                  |
                        0000   417 ;|                    |               |                                  |
                        0000   418 ;|                    |     CON 2     |                                  |
                        0000   419 ;|                    |               |                                  |
                        0000   420 ;|                    +---------------+                                  |
                        0000   421 ;|                                                                       |
                        0000   422 ;|                                                                       |
                        0000   423 ;|                                                                       |
                        0000   424 ;|                                                                       |
                        0000   425 ;+-----------------------------------------------------------------------+
```

```
                        0000   427          .SBTTL  "       L0011 & L0016 GATE ARRAY LOCATION"
                        0000   428 ;+-----------------------------------------------------------------------+
                        0000   429 ;|                                                                       |
                        0000   430 ;|                                                                       |
                        0000   431 ;|                                                                       |
                        0000   432 ;|                                                                       |
                        0000   433 ;|                                                                       |
                        0000   434 ;|                                                                       |
                        0000   435 ;|                                                                       |
                        0000   436 ;|                                                                       |
                        0000   437 ;|                                                                       |
                        0000   438 ;|                                                                       |
                        0000   439 ;|                                                                       |
                        0000   440 ;|                                                                       |
                        0000   441 ;|                                                                       |
                        0000   442 ;|                                                                       |
                        0000   443 ;|                                                                       |
                        0000   444 ;|                                                                       |
                        0000   445 ;|                                                                       |
                        0000   446 ;|                                                                       |
                        0000   447 ;|                                                                       |
                        0000   448 ;|                                                                       |
                        0000   449 ;|                                                                       |
                        0000   450 ;|                                                                       |
                        0000   451 ;|                                                                       |
                        0000   452 ;|                                                                       |
                        0000   453 ;|                                                                       |
                        0000   454 ;|                                                                       |
                        0000   455 ;|                                                                       |
                        0000   456 ;|                                                                       |
                        0000   457 ;|                                                                       |
                        0000   458 ;|                                                                       |
                        0000   459 ;|                                                                       |
                        0000   460 ;|                                                                       |
                        0000   461 ;|                                                                       |
                        0000   462 ;|                                                                       |
                        0000   463 ;|                                                                       |
                        0000   464 ;|                                                                       |
                        0000   465 ;|                                                                       |
                        0000   466 ;|                                                                       |
                        0000   467 ;|                                                                       |
                        0000   468 ;|                                                                       |
                        0000   469 ;|                                                                       |
                        0000   470 ;|                                                                       |
                        0000   471 ;|                                                                       |
                        0000   472 ;|                                                                       |
                        0000   473 ;|                                                                       |
                        0000   474 ;|                                                                       |
                        0000   475 ;|                                                                       |
                        0000   476 ;|                                                                       |
                        0000   477 ;|                                                                       |
                        0000   478 ;|                                                                       |
                        0000   479 ;|                                                                       |
                        0000   480 ;|                                                                       |
                        0000   481 ;|                                                                       |
                        0000   482 ;|                                                                       |
                        0000   483 ;+-----------------------------------------------------------------------+
```



```
                     0000   485          .SBTTL  ""
                     0000   486          .LIST   MC
                     0000   487          .LIBRARY        /UCOD$0:[VAX750]COMETMAC/
                     0000   488          .LIST   ME
                     0000   489          .NLIST  CND,MC
                     0000                .SBTTL  'EQUATED SYMBOLS
                     0000          ;
                     0000          ; RDM REGISTER DEFINITIONS:
                     0000          ;
                     0000          ; NOTE: THESE REGITER DEFINITIONS ARE OFFSET BY THE LOAD/START ADDRESS OF
                     0000          ;       THE PROGRAM. FOR THE SIMULATOR VERSION THIS IS 400(X) AND FOR THE
                     0000          ;       REAL VERSION IT IS 8400(X).
                     0000          ;
00007400             0000                DCS_DATA_REG    =       ^XF800 + HEAD
00007401             0000                DCS_ADDR_REG_WO =       ^XF801 + HEAD
00007402             0000                DCS_CTRL_REG    =       ^XF802 + HEAD
00007403             0000                DCS_CTRL_FILE   =       ^XF803 + HEAD
00007404             0000                ADDR_MATCH_LO   =       ^XF804 + HEAD
00007405             0000                ADDR_MATCH_HI   =       ^XF805 + HEAD
00007406             0000                STATUS_REG      =       ^XF806 + HEAD
                     0000          ;
00007410             0000                A_REG_BYTE_0    =       ^XF810 + HEAD
00007411             0000                A_REG_BYTE_1    =       ^XF811 + HEAD
00007412             0000                A_REG_BYTE_2    =       ^XF812 + HEAD
00007413             0000                A_REG_BYTE_3    =       ^XF813 + HEAD
00007414             0000                MD_REG_BYTE_0   =       ^XF814 + HEAD
00007415             0000                MD_REG_BYTE_1   =       ^XF815 + HEAD
00007416             0000                MD_REG_BYTE_2   =       ^XF816 + HEAD
00007417             0000                MD_REG_BYTE_3   =       ^XF817 + HEAD
00007418             0000                MH_REG_BYTE_0   =       ^XF818 + HEAD
00007419             0000                MH_REG_BYTE_1   =       ^XF819 + HEAD
0000741A             0000                MH_REG_BYTE_2   =       ^XF81A + HEAD
0000741B             0000                MH_REG_BYTE_3   =       ^XF81B + HEAD
0000741C             0000                CMI_CTRL_REG    =       ^XF81C + HEAD
0000741D             0000                MAINT_REG       =       ^XF81D + HEAD
0000741E             0000                RD_CTRL_REG     =       ^XF81E + HEAD
                     0000          ;
00007420             0000                FRONT_PNL_1     =       ^XF820 + HEAD
00007421             0000                FRONT_PNL_2     =       ^XF821 + HEAD
00007422             0000                CS_ADD_TRAP_LOW =       ^XF822 + HEAD
00007423             0000                CS_ADD_TRAP_HGH =       ^XF823 + HEAD
00007424             0000                CS_ADD_BUFF_LOW =       ^XF824 + HEAD
00007425             0000                CS_ADD_BUFF_HGH =       ^XF825 + HEAD
00007426             0000                BUFF_ADDR_LOW   =       ^XF826 + HEAD
00007427             0000                DCS_ADDR_REG_RO =       ^XF827 + HEAD
                     0000          ;
0000742C             0000                WD_REG_BYTE_0   =       ^XF82C + HEAD
0000742D             0000                WD_REG_BYTE_1   =       ^XF82D + HEAD
0000742E             0000                WD_REG_BYTE_2   =       ^XF82E + HEAD
0000742F             0000                WD_REG_BYTE_3   =       ^XF82F + HEAD
00007430             0000                WH_REG_BYTE_0   =       ^XF830 + HEAD
00007431             0000                WH_REG_BYTE_1   =       ^XF831 + HEAD
00007432             0000                WH_REG_BYTE_2   =       ^XF832 + HEAD
00007433             0000                WH_REG_BYTE_3   =       ^XF833 + HEAD
                     0000          ;
                     0000          ; RDM REGISTER NAMES AS DEFINED IN HARDWARE SPEC
                     0000          ;
```

```
00007400  0000              DCSDA=DCS_DATA_REG              ;Diagnostic control store data register
00007401  0000              DCSAD=DCS_ADDR_REG_WO           ;Diagnostic control store address reg
00007402  0000              DCSCR=DCS_CTRL_REG              ;Diagnostic control store control reg
00007403  0000              DCSCF=DCS_CTRL_FILE             ;Diagnostic control store control file
00007404  0000              CSAMR=ADDR_MATCH_LO             ;Control store address match register
00007406  0000              STATUS=STATUS_REG               ;Status register
00007410  0000              MAREG=A_REG_BYTE_0              ;Memory address register
00007414  0000              MDREG=MD_REG_BYTE_0             ;Memory data register
00007418  0000              MHREG=MH_REG_BYTE_0             ;Memory holding register
0000741C  0000              CMICTL=CMI_CTRL_REG             ;CMI control
0000741D  0000              MAIN32=MAINT_REG                ;32 Bit path maintenance control
0000741E  0000              RDCTRL=RD_CTRL_REG              ;Remote diagnosis control
00007420  0000              FRONT1=FRONT_PNL_1              ;Front panel readback 1
00007421  0000              FRONT2=FRONT_PNL_2              ;Front panel readback 2
00007422  0000              CSTRP=CS_ADD_TRAP_LOW           ;Control store address trap register
00007424  0000              CSBUF=CS_ADD_BUFF_LOW           ;Control store address buffer
00007426  0000              BADD1=BUFF_ADDR_LOW             ;Control store address trace readback
00007427  0000              CLKDCS=DCS_ADDR_REG_RO          ;Clock control & DCS address register
0000742C  0000              WDREG=WD_REG_BYTE_0             ;WBUS data register
00007430  0000              WHREG=WH_REG_BYTE_0             ;WBUS holding register
          0000          ;
          0000          ; RDM REGISTER BIT DEFINITIONS:
          0000          ;
          0000          ; DCS CONTROL REGISTER
00000001  0000                HALT_ON_MATCH   =       1
00000002  0000                CLEAR_CTRL_FILE =       2
00000004  0000                TRACE_ENABL     =       4
00000008  0000                PAR_CHK_ENABL   =       8
00000010  0000                PAR_STOP_ENABL  =       ^X10
00000020  0000                CLK_CTRL_0      =       ^X20    ; ACTIVE LOW
00000040  0000                CLK_CTRL_1      =       ^X40    ; ACTIVE LOW
00000080  0000                MICRO_ADDR_INH  =       ^X80
          0000            ;
          0000            ; FOLLOWING IS A FUNCTION TABLE OF THE CLOCK CONTROL BITS
          0000            ;
          0000                ;       CLK_CTRL 1,0    FUNCTION
          0000                ;           11            HALT
          0000                ;           01            SINGLE MICRO INSTRUCTION
          0000                ;           10            SINGLE TICK
          0000                ;           00            RUN
          0000          ;
          0000          ; DCS CONTROL FILE      ALL OF THESE BITS ARE ACTIVE LOW
          0000          ;
00000001  0000                V_BUS_STROBE    =       1
00000002  0000                DCS_ADDR_CLEAR  =       2
00000004  0000                STOP_CLOCK      =       4
00000008  0000                EN_TRACE_REG    =       8
00000010  0000                H_FROM_WBUS     =       ^X10
00000020  0000                WBUS_FROM_D     =       ^X20
00000040  0000                STROBE_CMI      =       ^X40
00000080  0000                SA_CLOCK        =       ^X80
          0000          ;
          0000          ; CS ADDRESS MATCH REGISTER (HIGH)
          0000          ;
00000040  0000                VBUS_SERIAL_IN  =       ^X40    ; ACTIVE LOW
00000080  0000                VBUS_CLOCK      =       ^X80
          0000          ;
```

```
                        0000    ; STATUS REGISTER
                        0000    ;
00000001                0000            VBUS_SERIAL_OUT =       1
00000002                0000            TRAP_OFF        =       2
00000008                0000            CMI_STATUS_0    =       8
00000010                0000            CMI_STATUS_1    =       ^X10
                        0000            ;
                        0000            ; THE CMI STATUS BITS ARE ENCODED AS FOLLOWS:
                        0000            ;
                        0000            ;       00 = NO RESPONSE
                        0000            ;       01 = UNCORRECTABLE ERROR
                        0000            ;       10 = CORRECTABLE ERROR
                        0000            ;       11 = NO ERRORS
00000020                0000            CMI_COMPLETE    =       ^X20
                        0000    ;
00000080                0000            CLOCK_STOPED    =       ^X80
                        0000    ;
                        0000    ; CMI CONTROL REGISTER
                        0000    ;
00000001                0000            CMI_CTRL_CLEAR  =       1
00000008                0000            CMI_GO          =       8
00000020                0000            CMI_WRITE       =       ^X20
00000040                0000            CMI_READ        =       ^X40
00000080                0000            SA_START_STOP   =       ^X80
                        0000    ;
                        0000    ; 32 BIT MAINTENANCE CONTROL REGISTER
                        0000    ;
00000001                0000            MAINT_TRAP_OFF  =       1
00000002                0000            MAINT_STB_INH   =       2
00000004                0000            MAINT_DCS_ENABL =       4
00000008                0000            MAINT_WB_TO_WH  =       8
00000010                0000            MAINT_WD_TO_WB  =       ^X10
00000020                0000            MAINT_CMI_TO_MH =       ^X20
00000040                0000            MAINT_MD_TO_CMI =       ^X40
00000080                0000            MAINT_A_TO_CMI  =       ^X80
                        0000    ;
                        0000    ; RD CONTROL REGISTER
                        0000    ;
00000008                0000            MASTER_HALT_EN  =       8
00000010                0000            VBUS_LOAD       =       ^X10
00000020                0000            TRAP_HALT_EN    =       ^X20
00000040                0000            DC_LOW          =       ^X40
00000080                0000            AC_LOW          =       ^X80
                        0000    ;
                        0000    ; FRONT PANEL REGISTER 1
                        0000    ;
00000001                0000            FP_BOOT_0       =       1
00000002                0000            FP_BOOT_1       =       2
00000004                0000            FP_START_0      =       4
00000008                0000            FP_START_1      =       8
00000010                0000            CPU_RUN         =       ^X10
00000020                0000            PARITY_ERROR    =       ^X20
                        0000    ;
                        0000    ; FRONT PANEL REGISTER 2
                        0000    ;
00000001                0000            REMOTE          =       1
00000002                0000            FAULT           =       2
```

```
00000004  0000                TEST                =       4
          0000        ;
          0000        ;
          0000        ;
00000040  0000                PB_INIT             =       ^X40    ; ACTIVE LOW
00000080  0000                FRONT_PNL_LOCK      =       ^X80
          0000        ;
          0000        ; DCS ADDRESS REGISTER READ ONLY
          0000        ;
00000040  0000                CLK_CTRL_0_RO       =       ^X40    ; ACTIVE HIGH
00000080  0000                CLK_CTRL_1_RO       =       ^X80    ; ACTIVE HIGH
          0000        ;
          0000        ; MISCELLANEOUS EQUATES:
          0000        ;
00000001  0000                BYTE                =       1
00000002  0000                WORD                =       2
00000004  0000                LONG                =       4
0000000A  0000                MICROWORD           =       10
00002C5E  0000                MONITOR_SIZE        =       ^X2C5E
00000400  0000                STACK_SIZE          =       1024
000007A2  0000                M$TEST_SIZE         =       14336 - STACK_SIZE - MONITOR_SIZE
          0000                                              ; MAXIMUM SIZE OF A TEST OVERLAY
000003E2  0000                M$TEST_SIZE_D       = M$TEST_SIZE - 960
          0000                                              ; MAXIMUM SIZE OF TEST OVERLAY WITH
          0000                                              ; 'DEBUG' ENABLED IN
00000001  0000                B                   =       BYTE
00000002  0000                W                   =       WORD
00000004  0000                L                   =       LONG
0000000A  0000                M                   =       MICROWORD
00000001  0000                HIGH                =       1
00000000  0000                LOW                 =       0
00000000  0000                ONERROR             =       0
00000001  0000                NOERROR             =       1
00000002  0000                ONEQUAL             =       2
00000003  0000                NOTEQUAL            =       3
00000010  0000                MAX_ARGUMENTS       =       16      ; BYTE LENGTH OF LARGEST PSEUDO OP
00000036  0000                DCS_SCRATCH_ADR     =       54      ; DCS SCRATCH ADDRESS START
000032FC  0000                ERROR_LOG_ADR       =       ^XB6FC + HEAD
          0000                                              ; START ADDRESS OF MEMORY ERROR LOG
          0000    491         .NLIST  ME
          0000    492         .LIST   MC
          0000    493         .SBTTL  ""
          0000    494         .NLIST  MC      ; SHUT-OFF LISTING FOR FIRST 'NEWTEST' CALL
```

```
                        001C        .SBTTL  "TEST   0CE     CM Odd Address BUT Test
                        001C      3 ;*******************************************************************************
                        001C      4 ;++
                        001C      5 ;
                        001C      6 ; FUNCTIONAL DESCRIPTION
                        001C      7 ;
                        001C      8 ;       This test examines the BUT/CM.ODD.ADD micro-order for proper operation.
                        001C      9 ;       The BUT/CM.ODD.ADD micro-order will force a one into CSA<0> when VA<0>
                        001C     10 ;       is a one and D Size is Byte.
                        001C     11 ;
                        001C     12 ;
                        001C     13 ; TEST ALGORITHM
                        001C     14 ;
                        001C     15 ;       To fully test the BUT/CM.ODD.ADD micro-order all four combinations of
                        001C     16 ;       VA<0> and D Size equal to byte should be tried. This test has been
                        001C     17 ;       divided into four subtests. Each trys a different combination of VA<0>
                        001C     18 ;       and D Size equal to byte.
                        001C     19 ;
                        001C     20 ;       The test is implemented by the following algorithm:
                        001C     21 ;
                        001C     22 ;       Subtest 1
                        001C     23 ;
                        001C     24 ;       This subtest loads the VA with 0, the IR with a MOVB instruction and
                        001C     25 ;       expects a 0 in CSA<0> during the BUT/CM.ODD.ADD micro-branch.
                        001C     26 ;
                        001C     27 ;        1. Initialize the CPU.
                        001C     28 ;        2. Open the window to the signature analyzer.
                        001C     29 ;        3. Execute the following micro-code to set up the test:
                        001C     30 ;                1. Place the CPU into compatibility mode.
                        001C     31 ;                2. Disable console CTRL-P interrupts from occuring.
                        001C     32 ;                3. Zero the VA.
                        001C     33 ;                4. Write the test opcodes to cache for XB0.
                        001C     34 ;                5. Increment the VA.
                        001C     35 ;                6. Write zeroes into the next cache location for XB1.
                        001C     36 ;                7. Zero the VA.
                        001C     37 ;                8. Zero the PC to initiate instruction prefetch.
                        001C     38 ;                9. Load the IR and the OSR with the first test opcode.
                        001C     39 ;        4. Prepare the RDM to expect any control store address.
                        001C     40 ;        5. Loop on error from here.
                        001C     41 ;        6. Execute a BUT/CM.ODD.ADD micro-branch with 1800 in the NEXT field.
                        001C     42 ;        7. Using a mask of 3FFF compare the Control Store address TRaP
                        001C     43 ;           register (CSTRP) with the expected result of 1800.
                        001C     44 ;        8. Execute micro-code to write the IR and OSR into RTEMP0 and to write
                        001C     45 ;           the current value of the VA to RTEMP1:
                        001C     46 ;                10. Write the IR into the MDR.
                        001C     47 ;                11. Write an eight into the P-LATCH on the rotator.
                        001C     48 ;                12. Shift the MDR left 8 bits and store the result into RTEMP0.
                        001C     49 ;                13. Write the OSR into the MDR.
                        001C     50 ;                14. OR the MDR with RTEMP0 and store the result in RTEMP0.
                        001C     51 ;                15. Write the VA into RTEMP1.
                        001C     52 ;        9. If the address in the Control Store address TRaP register (CSTRP)
                        001C     53 ;           is not 1800, signal an error and dump out RTEMP0 and RTEMP1.
                        001C     54 ;
                        001C     55 ;       Subtest 2
                        001C     56 ;
                        001C     57 ;       This subtest loads the VA with 1, keeps the MOVB instruction in the IR,
                        001C     58 ;       and expects a 0 in CSA<0> during the BUT/CM.ODD.ADD micro-branch.
```

```
001C     59 ;
001C     60 ;       10. Execute the micro-code to load VA<0> with a one.
001C     61 ;       11. Prepare the RDM to expect any control store address.
001C     62 ;       12. Loop on error from here.
001C     63 ;       13. Execute the BUT/CM.ODD.ADD branch using the same opcode.
001C     64 ;       14. Do a masked comparison of the CSTRP register against 1800.
001C     65 ;       15. Write the IR and OSR to RTEMP0 and the VA to RTEMP1.
001C     66 ;       16. If the control store address was not 1800 signal an error.
001C     67 ;
001C     68 ;       Subtest 3
001C     69 ;
001C     70 ;       This subtest loads the VA with 0, loads a MOV instruction into the IR,
001C     71 ;       and expects a 0 in CSA<0> during the BUT/CM.ODD.ADD micro-branch.
001C     72 ;
001C     73 ;       17. Execute micro-code to zero the VA and load the IR with a MOV inst.
001C     74 ;       18. Prepare the RDM to expect any control store address.
001C     75 ;       19. Loop on error form here.
001C     76 ;       20. Execute the BUT/CM.ODD.ADD branch.
001C     77 ;       21. Do a masked comparison of the CSTRP register against 1800.
001C     78 ;       22. Write the IR and OSR to RTEMP0 and the VA to RTEMP1.
001C     79 ;       23. If the control store address was not 1800 signal an error.
001C     80 ;
001C     81 ;       Subtest 4
001C     82 ;
001C     83 ;       This subtest loads the VA with 1, keeps the MOV instruction in the IR,
001C     84 ;       and expects a 1 in CSA<0> during the BUT/CM.ODD.ADD micro-branch.
001C     85 ;
001C     86 ;       24. Execute micro-code to put a one in VA<0>.
001C     87 ;       25. Prepare the RDM to expect any control store address.
001C     88 ;       26. Loop on error from here.
001C     89 ;       27. Execute the BUT/CM.ODD.ADD branch on the MOV instruction.
001C     90 ;       28. Do a masked comparison of the CSTRP register against 1801.
001C     91 ;       29. Write the IR and OSR to RTEMP0 and the VA to RTEMP1.
001C     92 ;       30. If the control store address was not 1801 signal an error.
001C     93 ;       31. Close the window to the signature analyzer.
001C     94 ;       32. Execute micro-code to restore the previous CPU state:
001C     95 ;               16. Enable CTRL-P interrupts.
001C     96 ;               17. Zero the PSL to restore native mode.
001C     97 ;       33. Skip to next test.
001C     98 ;
001C     99 ; TEST DATA
001C    100 ;
001C    101 ;       This table shows the value of the IR, OSR and the VA for each subtest.
001C    102 ;
001C    103 ;       Subtest 1       IR/OSR = ^X9000 (MOVB)  VA = 0
001C    104 ;       Subtest 2       IR/OSR = ^X9000 (MOVB)  VA = 1
001C    105 ;       Subtest 3       IR/OSR = ^X1000 (MOV)   VA = 0
001C    106 ;       Subtest 4       IR/OSR = ^X1000 (MOV)   VA = 1
001C    107 ;
001C    108 ;       This table shows the initial value and function of the RTEMPS used.
001C    109 ;
001C    110 ;       RTEMP0 = 0              Current opcode under execution
001C    111 ;       RTEMP1 = 0              Current value of the VA
001C    112 ;       RTEMP2 = ^X00000001     Odd VA
001C    113 ;       RTEMP3 = ^X10009000     Opcodes for XB0
001C    114 ;       RTEMP4 = ^X80000000     Data to place CPU into compatibility mode
001C    115 ;       RTEMP5 = ^X00C00000     Address of status and control bit in CRAR
```

```
                        001C    116 ;       RTEMP6 = ^X00400000     Data to disable control-p interrupts
                        001C    117 ;       RTEMP7 = ^X00900000     Data to enable control-p interrupts
                        001C    118 ;
                        001C    119 ; LOGICAL DESCRIPTION
                        001C    120 ;
                        001C    121 ;       In this test the value of bit zero of the control store address
                        001C    122 ;       (CSA<0>) is determined by three things; discrete logic on the DPM, the
                        001C    123 ;       IRD gate array, and the PHB gate array. CSA<0> is generated by an in-
                        001C    124 ;       verter that sits on the output of an 8 to 1 multiplexer. The mux is
                        001C    125 ;       enabled by the signal ENABLE BUT 4XXX, which is generated by a 1 of 8
                        001C    126 ;       decoder whose select lines depend upon bits 5:3 of the BUT field. For
                        001C    127 ;       the BUT/CM.ODD.ADD micro-order, input D1 of the mux is selected. This
                        001C    128 ;       input is the AND of WBUS<0> and D Size equal to byte. For this test to
                        001C    129 ;       work properly, the VA must be driven onto the WBUS during the BUT.
                        001C    130 ;
                        001C    131 ; ASSUMPTIONS
                        001C    132 ;
                        001C    133 ;       This test assumes the following logic has been tested prior to this
                        001C    134 ;       test. The VA, PC, the RTEMP's, and the IR and OSR.
                        001C    135 ;
                        001C    136 ; ERROR DESCRIPTION
                        001C    137 ;
                        001C    138 ;       On error this test will call out the IRD and PHB gate arrays and the
                        001C    139 ;       following data:
                        001C    140 ;
                        001C    141 ;       EXPECTED CONTROL STORE ADDRESS
                        001C    142 ;       RECEIVED CONTROL STORE ADDRESS
                        001C    143 ;       THE CURRENT CONTENTS OF THE IR AND OSR
                        001C    144 ;       THE CURRENT VALUE OF THE VA
                        001C    145 ;
                        001C    146 ;--
                        001C    147 ;*****************************************************************************
                        001C    148
                        001C    149         SUBTEST         1
                        001F
                        001F        ;////////////////////////////////////////////////////////////////////////////
                        001F    150 ;+
                        001F    151 ; This subtest loads a MOVB instruction into the IR, zeros the VA and executes
                        001F    152 ; a BUT/CM.ODD.ADD. It then expects to see 1800 in the control store address
                        001F    153 ; trap register. (CSTRP)
                        001F    154 ;-
                        001F    155         INITIALIZE                              ; INITIALIZE THE CPU
                        0021    156         FETCH           3                       ; EXECUTE MICRO-CODE TO SET UP
                        0026    157         BRSTCLK         <^X10>                  ; THIS AND SUBSEQUENT SUBTESTS
                        002A    158         EXPUTRAP                                ; EXPECT ANY CS ADDRESS
                        002C    159         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        002E    160         FETCH           0                       ; EXECUTE THE BUT/CM.ODD.ADD
                        0033    161         BRSTCLK         2                       ; ...
                        0037    162         CMPREGMSK       CSTRP,1$,,3$,,W         ; COMPARE CSTRP AGAINST 1800
                        0043    163         FETCH           <^X18>                  ; READ BACK THE IR AND OSR AND
                        0048    164         BRSTCLK         <^X1F>,INH              ; WRITE THEM TO RTEMP0
                        004C    165         IFERROR         2,<<IRD><PHB>>          ; IF CSTRP .NE. 1800 THEN ERROR
                        0053    166
                        0053    167         SUBTEST         2
                        0056
                        0056        ;////////////////////////////////////////////////////////////////////////////
                        0056    168 ;+
```

```
                        0056   169 ; This subtest loads VA<0> with a one and leaves the MOVB instruction in the
                        0056   170 ; IR. It then reexecutes the BUT/CM.ODD.ADD expecting CSTRP to equal 1800.
                        0056   171 ;-
                        0056   172         FETCH           <^X11>                  ; LOAD A 1 IN VA<0>
                        005B   173         BRSTCLK         <^X13>                  ; ...
                        005F   174         EXPUTRAP                                ; EXPECT ANY ADDRESS
                        0061   175         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        0063   176         FETCH           0                       ; EXECUTE THE BUT/CM.ODD.ADD ON
                        0068   177         BRSTCLK         2                       ; THE SAME OPCODE
                        006C   178         CMPREGMSK       CSTRP,1$,,3$,,W         ; COMPARE CSTRP AGAINST 1800
                        0078   179         FETCH           <^X18>                  ; READ BACK THE IR AND OSR AND
                        007D   180         BRSTCLK         <^X1F>,INH              ; WRITE THEM TO RTEMP0
                        0081   181         IFERROR         2,<<IRD><PHB>>          ; IF CSTRP .NE. 1800 THEN ERROR
                        0088   182
                        0088   183         SUBTEST         3
                        008B
                        008B       ;////////////////////////////////////////////////////////////////////////////////
                        008B   184 ;+
                        008B   185 ; This subtest loads a MOV instruction into the IR, zeros the VA and executes
                        008B   186 ; a BUT/CM.ODD.ADD. It then expects to see 1800 in CSTRP.
                        008B   187 ;-
                        008B   188         FETCH           <^X14>                  ; ZERO VA AND LOAD THE OPCODE
                        0090   189         BRSTCLK         <^X17>                  ; FOR THE MOV INSTRUCTION
                        0094   190         EXPUTRAP                                ; EXPECT ANY CS ADDRESS
                        0096   191         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        0098   192         FETCH           0                       ; EXECUTE THE BUT/CM.ODD.ADD
                        009D   193         BRSTCLK         2                       ; ...
                        00A1   194         CMPREGMSK       CSTRP,1$,,3$,,W         ; COMPARE CSTRP AGAINST 1800
                        00AD   195         FETCH           <^X18>                  ; WRITE IR AND OSR TO RTEMP0
                        00B2   196         BRSTCLK         <^X1F>,INH              ; AND WRITE VA TO RTEMP1
                        00B6   197         IFERROR         2,<<IRD><PHB>>          ; IF CSTRP .NE. 1800 THEN ERROR
                        00BD   198
                        00BD   199         SUBTEST         4
                        00C0
                        00C0       ;////////////////////////////////////////////////////////////////////////////////
                        00C0   200 ;+
                        00C0   201 ; This subtest loads VA<0> with a one and leaves the MOV instruction in the IR.
                        00C0   202 ; It then reexecutes the BUT/CM.ODD.ADD expecting CSTRP to equal 1801.
                        00C0   203 ;-
                        00C0   204         FETCH           <^X11>                  ; LOAD A 1 IN VA<0>
                        00C5   205         BRSTCLK         <^X13>                  ; ...
                        00C9   206         EXPUTRAP                                ; EXPECT ANY CS ADDRESS
                        00CB   207         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        00CD   208         FETCH           0                       ; EXECUTE BUT/CM.ODD.ADD
                        00D2   209         BRSTCLK         2                       ; ...
                        00D6   210         CMPREGMSK       CSTRP,2$,,3$,,W         ; COMPARE CSTRP AGAINST 1801
                        00E2   211         FETCH           <^X18>                  ; WRITE IR AND OSR TO RTEMP0
                        00E7   212         BRSTCLK         <^X1F>,INH              ; AND WRITE VA TO RTEMP1
                        00EB   213         IFERROR         2,<<IRD><PHB>>          ; IF CSTRP .NE. 1801 THEN ERROR
                        00F2   214
                        00F2   215         FETCH           <^X20>                  ; ENABLE CTRL-P INTERRUPTS AND
                        00F7   216         BRSTCLK         <^X23>                  ; RETURN CPU TO NATIVE MODE
                        00FB   217         SKIP                                    ; END OF TEST
                        0102   218
                        0102   219 BEGIN_TEST_DATA
                        0102
                        0102       ;-----------------------------------------------------------------------------
```

```
                                0102   220
                        1800    0102   221 1$:     .WORD   ^X1800          ; EXPECTED ADDRESS FOR SUBTESTS 1 - 3
                        1801    0104   222 2$:     .WORD   ^X1801          ; EXPECTED ADDRESS FOR SUBTEST 4
                        3FFF    0106   223 3$:     .WORD   ^X3FFF          ; MASK FOR ADDRESS COMPARISON
                                0108   224
                                0108   225 BEGIN_CPU_DATA
                                0002   226
                                0002   227 DCS_DATA
                                0001   228
U 00. 7700,C800,0364,0300,0470,1801     0001   229 ;% 00   NOP                                     ; GET CLOCK UP TO SPEED
U 01. 7700,081B,5924,8700,0470,1800     000C   233 ;% 01   NEXT/1800,BUT/CM.ODD.ADD,WB_M[VA]       ; BRANCH NEEDS VA<0> ON WB
U 02. 7300,4800,0364,0300,0470,1803     0017   237 ;% 02   STOP.CLOCK                              ; HALT
                                0022   241
U 03. 7700,C800,0364,0300,0470,1804     0022   242 ;% 03   NOP                                     ; GET CLOCK UP TO SPEED
U 04. 7700,8800,5BE4,0310,0070,1805     002D   246 ;% 04   WB_R[TEMP4],CCPSL/PSL_WB.CCBR_ALUS      ; PUT CPU INTO COMP. MODE
U 05. 7700,C800,5BE4,0314,2470,1806     0038   250 ;% 05   WCTRL/LOADCRAR,WB_R[TEMP5]              ; ADDRESS OF S&C BIT IN CRAR
U 06. 7700,C800,5BE4,0318,2C70,1807     0043   254 ;% 06   WCTRL/CONWRITE,WB_R[TEMP6]              ; DISABLE CTRL-P INTERRUPTS
U 07. 7700,C800,5BE4,03D8,4A70,1808     004E   258 ;% 07   VA_R[ZERO]                              ; ZERO VA
U 08. 7701,0800,5BE4,020C,5590,1809     0059   262 ;% 08   WRITE.LONG R[TEMP3]                     ; WRITE TEST OPCODES TO CACHE
U 09. 7701,0800,5BE4,020C,55F0,180A     0064   266 ;% 09   WDR_UNROT(R[TEMP3]),BUS/PRB.RD,SIZE[LONG]
                                006F   270                                                     ; HOLD FOR 2ND CYCLE
U 0A. 7700,4800,0364,0300,4470,180B     006F   271 ;% 0A   VA_VA+4                               ; INCREMENT THE VA
U 0B. 7701,0800,5BE4,02D8,5590,180C     007A   275 ;% 0B   WRITE.LONG R[ZERO]                    ; VALIDATE THE SECOND CACHE
U 0C. 7701,8800,5BE4,02D8,55F0,180D     0085   279 ;% 0C   WDR_UNROT(R[ZERO]),BUS/PRB.RD,SIZE[LONG]
                                0090   283                                                     ; LOCATION FOR XB1
U 0D. 7700,C800,5BE4,03D8,4A70,180E     0090   284 ;% 0D   VA_R[ZERO]                              ; ZERO VA
U 0E. 7700,C800,5BE4,03D8,4870,180F     009B   288 ;% 0E   PC_R[ZERO]                              ; ZERO PC AND LOAD XB'S
U 0F. 7700,C800,0364,1700,0470,1810     00A6   292 ;% 0F   BUT/IRD1TST                            ; LOAD IR WITH MOVB INSTRUCTION
U 10. 7300,4800,0364,0300,0470,1811     00B1   296 ;% 10   STOP.CLOCK                             ; HALT
                                00BC   300
U 11. 7700,4800,0364,0300,0470,1812     00BC   301 ;% 11   NOP                                     ; GET CLOCK UP TO SPEED
U 12. 7700,8800,5BE4,0308,4A70,1813     00C7   305 ;% 12   VA_R[TEMP2]                             ; LOAD A ONE IN VA<0>
U 13. 7300,4800,0364,0300,0470,1814     00D2   309 ;% 13   STOP.CLOCK                             ; HALT
                                00DD   313
U 14. 7700,C800,0364,0300,0470,1815     00DD   314 ;% 14   NOP                                     ; GET CLOCK UP TO SPEED
U 15. 7700,C800,5BE4,03D8,4A70,1816     00E8   318 ;% 15   VA_R[ZERO]                              ; ZERO THE VA
U 16. 7700,4800,0364,1700,0470,1817     00F3   322 ;% 16   BUT/IRD1TST                            ; LOAD IR WITH MOV INSTRUCTION
U 17. 7300,4800,0364,0300,0470,1818     00FE   326 ;% 17   STOP.CLOCK                             ; HALT
                                0109   330
U 18. 7700,C800,0364,0300,0470,1819     0109   331 ;% 18   NOP                                     ; GET CLOCK UP TO SPEED
U 19. 7700,4800,0364,0300,5670,181A     0114   335 ;% 19   WCTRL/MDR_IR                            ; READ IR TO MDR
U 1A. 7700,D800,EF64,0304,0470,181B     011F   339 ;% 1A   PL_[8]                                 ; PUT AN 8 INTO THE P LATCH
U 1B. 7700,8852,8770,0300,0470,181C     012A   343 ;% 1B   R[TEMP0]_M[MDR].RL.P                   ; RTEMP0 = OPCODE .RL. 8 BITS
U 1C. 7700,4800,0364,0300,5E70,181D     0135   347 ;% 1C   CCPSL/MDR_OSR.CCBR_BRATST              ; READ OSR TO MDR
U 1D. 7700,4852,0024,0300,0470,181E     0140   351 ;% 1D   R[TEMP0]_M[MDR].OR.RB                  ; RTEMP0 = INSTRUCTION OPCODE
U 1E. 7700,885B,5924,0304,0470,181F     014B   355 ;% 1E   R[TEMP1]_VA                            ; RTEMP1 = VA
U 1F. 7300,C800,0364,0300,0470,1820     0156   359 ;% 1F   STOP.CLOCK                             ; HALT
                                0161   363
U 20. 7700,4800,0364,0300,0470,1821     0161   364 ;% 20   NOP                                     ; GET CLOCK UP TO SPEED
U 21. 7700,0800,5BE4,031C,2C70,1822     016C   368 ;% 21   WCTRL/CONWRITE,WB_R[TEMP7]              ; ENABLE CTRL-P INTERRUPTS
U 22. 7700,4800,5BE4,03D8,0070,1823     0177   372 ;% 22   WB_R[ZERO],CCPSL/PSL_WB.CCBR_ALUS      ; PUT CPU BACK INTO NATIVE MODE
U 23. 7300,4800,0364,0300,0470,1824     0182   376 ;% 23   STOP.CLOCK                             ; HALT
                                018D   380
                                018D   381 RTEMP_DATA
                                0001   382
                        00000000        0001   383         .LONG   0               ; OPCODE UNDER EXECUTION
                        00000000        0005   384         .LONG   0               ; CURRENT VALUE OF THE VA
```

```
00000001  0009   385          .LONG   ^X00000001      ; ODD VA
10009000  000D   386          .LONG   ^X10009000      ; OPCODES FOR XB0
80000000  0011   387          .LONG   ^X80000000      ; DATA TO PLACE CPU INTO COMP. MODE
00C00000  0015   388          .LONG   ^X00C00000      ; ADDRESS OF STATUS AND CONTROL BIT IN CRAR
00400000  0019   389          .LONG   ^X00400000      ; DATA TO DISABLE CTRL-P INTERRUPT
00900000  001D   390          .LONG   ^X00900000      ; DATA TO ENABLE CTRL-P INTERRUPT
          0021   391
          0021   392 END_CPU_DATA
          0108   393
          0108   394 END_TEST_DATA
          0108
          0108       ;-------------------------------------------------------------------------
          0108   395
          0108   396 ENDTEST
```

```
0019          .SBTTL  "TEST   OCF     CM IR2.TO.0 BUT Test
0019    398 ;*****************************************************************************
0019    399 ;++
0019    400 ;
0019    401 ; FUNCTIONAL DESCRIPTION
0019    402 ;
0019    403 ;       This test examines the BUT/IR2.TO.0 micro-branch while the CPU is in
0019    404 ;       compatibility mode. This will insure that the micro-branch is still
0019    405 ;       done on IR<2:0> even when the CPU is in compatibility mode.
0019    406 ;
0019    407 ; TEST ALGORITHM
0019    408 ;
0019    409 ;       This test is implemented by running through all eight combinations of
0019    410 ;       IR<2:0>. The steps required are:
0019    411 ;
0019    412 ;        1. Initialize the CPU.
0019    413 ;        2. Open the window to the signature analyzer.
0019    414 ;        3. Tell the RDM to expect any control store address.
0019    415 ;        4. Execute micro-code to put the CPU into compatibility mode and to
0019    416 ;           disable console Control-P interrupts.
0019    417 ;        5. Initialize the WD register to zero.
0019    418 ;        6. Loop for all combinations of IR<2:0>. (8 iterations)
0019    419 ;        7. Load the WD register with a model opcode indexed by the loop count.
0019    420 ;        8. Execute the following micro-code to set up the test:
0019    421 ;                1. Write the model opcode from the WD register to RTEMP0.
0019    422 ;                2. Zero the VA.
0019    423 ;                3. Write the model opcode from RTEMP0 to cache.
0019    424 ;                4. Increment the VA.
0019    425 ;                5. Zero the next cache location for XB1.
0019    426 ;                6. Zero the PC to initiate instruction prefetch.
0019    427 ;                7. Load the IR and OSR using a BUT/IRD1TST micro-order.
0019    428 ;        9. Loop on error from here.
0019    429 ;       10. Execute the BUT/IR2.TO.0 micro-branch with 1800 in the NEXT field.
0019    430 ;       11. Compare (masked) the expected and received control store addresses.
0019    431 ;       12. Execute the micro-code to place the IR and OSR into RTEMP0.
0019    432 ;       13. If the addresses are not equal signal and error and dump RTEMP0.
0019    433 ;       14. Continue the loop until every combination of IR<2:0> is tried.
0019    434 ;       15. Execute the micro-code to put CPU back into native mode and to
0019    435 ;           enable console Control-P interrupts.
0019    436 ;       16. Close the window to the signature analyzer.
0019    437 ;       17. End of test.
0019    438 ;
0019    439 ; TEST DATA
0019    440 ;
0019    441 ;       This table shows the opcodes used to implement the test. The patterns
0019    442 ;       form a count pattern based on IR<2:0>.
0019    443 ;
0019    444 ;       Loop    Opcode  CS Address      IR<2:0>
0019    445 ;
0019    446 ;         1.     1000     1800           000
0019    447 ;         2.     1100     1801           001
0019    448 ;         3.     1200     1802           010
0019    449 ;         4.     1300     1803           011
0019    450 ;         5.     1400     1804           100
0019    451 ;         6.     1500     1805           101
0019    452 ;         7.     1600     1806           110
0019    453 ;         8.     1700     1807           111
```

```
                        0019    454 ;
                        0019    455 ;       This table shows the initial contents of the RTEMP registers used in
                        0019    456 ;       the test.
                        0019    457 ;
                        0019    458 ;               RTEMP0 = Current contents of the IR and OSR
                        0019    459 ;               RTEMP1 = Data to place CPU into compatibility mode
                        0019    460 ;               RTEMP2 = Address of status and control bit in CRAR
                        0019    461 ;               RTEMP3 = Data to disable console Control-P interrupts
                        0019    462 ;               RTEMP4 = Data to enable console Control-P interrupts
                        0019    463 ;
                        0019    464 ; LOGICAL DESCRIPTION
                        0019    465 ;
                        0019    466 ;       The logic involved in this test is internal to the IRD gate array.
                        0019    467 ;       Bits <2:0> of the IR are ORed directly onto the control store address
                        0019    468 ;       bits <2:0>, by gates inside IRD. The control signals IRD ADD CTL 1 and
                        0019    469 ;       IRD ADD CTL 0 are both required to be high for the IRD chip to perform
                        0019    470 ;       this function. These two control signals are generated from the micro-
                        0019    471 ;       word by PHB.
                        0019    472 ;
                        0019    473 ; ASSUMPTIONS
                        0019    474 ;
                        0019    475 ;       This test assumes the following logic is working; the VA, PC, RTEMPS,
                        0019    476 ;       ALU, ROTATOR, the IR and OSR, and the logic used to do an instruction
                        0019    477 ;       prefetch.
                        0019    478 ;
                        0019    479 ; ERROR DESCRIPTION
                        0019    480 ;
                        0019    481 ;       On error this test will call out the IRD and PHB gate arrays.
                        0019    482 ;       The following information will also by typed out:
                        0019    483 ;
                        0019    484 ;       EXPECTED CONTROL STORE ADDRESS
                        0019    485 ;       RECEIVED CONTROL STORE ADDRESS
                        0019    486 ;       LOOP COUNT
                        0019    487 ;       THE CURRENT CONTENTS OF THE IR AND OSR
                        0019    488 ;
                        0019    489 ;--
                        0019    490 ;*****************************************************************************
                        0019    491 ;/////////////////////////////////////////////////////////////////////////////
                        0019    492
                        0019    493         INITIALIZE                              ; INITIALIZE THE CPU
                        001B    494         FETCH           <^X15>                  ; PUT CPU INTO COMP. MODE AND
                        0020    495         BRSTCLK         <^X19>                  ; DISABLE CTRL-P INTERRUPTS
                        0024    496         LOADREG         WDREG,1$,,L             ; INITIALIZE THE WDREG TO ZERO
                        002C    497         LOOP            I,1,8                   ; LOOP FOR ALL BITS OF IR<2:0>
                        0033    498         LOADREG         WDREG,2$,I,W            ; LOAD WDREG WITH MODEL OPCODE
                        003B    499         FETCH           3                       ; LOAD IR AND OSR WITH OPCODE
                        0040    500         BRSTCLK         <^X0D>                  ; ...
                        0044    501         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        0046    502         FETCH           0                       ; EXECUTE THE BUT/IR2.TO.0
                        004B    503         BRSTCLK         2                       ; ...
                        004F    504         CMPREGMSK       CSTRP,3$,I,4$,,W        ; IS THE CS ADDRESS OK?
                        005B    505         FETCH           <^X0E>                  ; READ BACK THE IR AND OSR AND
                        0060    506         BRSTCLK         <^X14>,INH              ; PUT THEM INTO RTEMP0
                        0064    507         IFERROR         1,<<IRD><PHB>>          ; IF BAD ADDRESS SIGNAL ERROR
                        006B    508         ENDLOOP         I                       ; CONTINUE LOOP FOR ALL BITS
                        006E    509         FETCH           <^X1A>                  ; RESTORE CPU TO NATIVE MODE
                        0073    510         BRSTCLK         <^X1D>                  ; AND ENABLE CTRL-P INTERRUPTS
```

```
                                0077   511          SKIP                                    ; END OF TEST
                                007E   512
                                007E   513 BEGIN_TEST_DATA
                                007E
                                007E       ;-------------------------------------------------------------------
                                007E   514
                    00000000    007E   515 1$:      .LONG   0       ; ZERO TO INITIALIZE THE WD REGISTER
                                0082   516 ;
                                0082   517 ; Test opcodes
                                0082   518 ;
                                0082   519 ; Here are the test opcodes used to test IR<2:0>. The comments to the right
                                0082   520 ; indicate the value of the loop count for each opcode.
                                0082   521 ;
 1600 1500 1400 1300 1200 1100 1000   0082   522 2$:      .WORD   ^X1000,^X1100,^X1200,^X1300,^X1400,^X1500,^X1600 ; 01 - 07
                            1700    0090   523          .WORD   ^X1700                                            ; 08
                                0092   524 ;
                                0092   525 ; Control Store Addresses
                                0092   526 ;
                                0092   527 ; This is the table of expected control store addresses.
                                0092   528 ;
 1806 1805 1804 1803 1802 1801 1800   0092   529 3$:      .WORD   ^X1800,^X1801,^X1802,^X1803,^X1804,^X1805,^X1806 ; 01 - 07
                            1807    00A0   530          .WORD   ^X1807                                            ; 08
                                00A2   531
                            3FFF    00A2   532 4$:      .WORD   ^X3FFF  ; MASK FOR ADDRESS COMPARISON
                                00A4   533
                                00A4   534
                                00A4   535 BEGIN_CPU_DATA
                                0002   536
                                0002   537 DCS_DATA
                                0001   538
U 00, 7700.C800.0364.0300.0470.1801   0001   539 ;% 00   NOP                                     ; GET CPU CLOCK UP TO SPEED
U 01, 7700.0800.0364.6700.0470.1800   000C   543 ;% 01   NEXT/1800,BUT/IR.2TOO                  ; EXECUTE THE BRANCH TEST
U 02, 7300.4800.0364.0300.0470.1803   0017   547 ;% 02   STOP.CLOCK                             ; HALT
                                0022   551
U 03, 7700.C800.0364.0300.0470.1804   0022   552 ;% 03   NOP                                     ; GET CPU CLOCK UP TO SPEED
U 04, 5700.0840.0364.0300.0470.1805   002D   556 ;% 04   R[TEMP0]_RDM                           ; WRITE OPCODE TO RTEMP0
U 05, 7700.4800.5BE4.03D8.4A70.1806   0038   560 ;% 05   VA_R[ZERO]                            ; ZERO THE VA
U 06, 7701.8800.5BE4.0200.5590.1807   0043   564 ;% 06   WRITE.LONG R[TEMP0]                   ; WRITE TEST OPCODE TO CACHE
U 07, 7701.8800.5BE4.0200.55F0.1808   004E   568 ;% 07   WDR_UNROT(R[TEMP0]),BUS/PRB.RD,SIZE[LONG]
                                0059   572                                                   ; ...
U 08, 7700.C800.0364.0300.4470.1809   0059   573 ;% 08   VA_VA+4                               ; INCREMENT THE VA
U 09, 7701.0800.5BE4.02D8.5590.180A   0064   577 ;% 09   WRITE.LONG R[ZERO]                    ; ZERO NEXT CACHE LOCATION FOR
U 0A, 7701.8800.5BE4.02D0.55F0.180B   006F   581 ;% 0A   WDR_UNROT(R[ZERO]),BUS/PRB.RD,SIZE[LONG]                ; XB1
U 0B, 7700.C800.5BE4.03D8.4070.180C   007A   585 ;% 0B   PC_R[ZERO]                             ; INITIATE INSTRUCTION PREFETCH
U 0C, 7700.C800.0364.1700.0470.180D   0085   589 ;% 0C   BUT/IRD1TST                           ; LOAD THE IR AND OSR
U 0D, 7300.C800.0364.0300.0470.180E   0090   593 ;% 0D   STOP.CLOCK                             ; HALT
                                009B   597
U 0E, 7700.4800.0364.0300.0470.180F   009B   598 ;% 0E   NOP                                     ; GET THE CLOCK UP TO SPEED
U 0F, 7700.4800.0364.0300.5670.1810   00A6   602 ;% 0F   WCTRL/MDR_IR                           ; WRITE THE IR TO THE MDR
U 10, 7700.D800.EF64.0304.0470.1811   00B1   606 ;% 10   PL_[8]                                 ; PUT AN 8 INTO THE P-LATCH
U 11, 7700.0852.8770.0300.0470.1812   00BC   610 ;% 11   R[TEMP0]_M[MDR].RL.P                  ; RTEMP0 = IR .RL. 8 BITS
U 12, 7700.C800.0364.0300.5E70.1813   00C7   614 ;% 12   CCPSL/MDR_OSR.CCBR_BRATST            ; WRITE THE OSR TO THE MDR
U 13, 7700.4852.0024.0300.0470.1814   00D2   618 ;% 13   R[TEMP0]_M[MDR].OR.RB               ; RTEMP0 = INSTRUCTION OPCODE
U 14, 7300.C800.0364.0300.0470.1815   00DD   622 ;% 14   STOP.CLOCK                             ; HALT
                                00E8   626
U 15, 7700.C800.0364.0300.0470.1816   00E8   627 ;% 15   NOP                                     ; GET THE CLOCK UP TO SPEED
U 16, 7700.8800.5BE4.0304.0070.1817   00F3   631 ;% 16   WB_R[TEMP1],CCPSL/PSL_WB.CCBR_ALUS     ; PUT CPU INTO COMP. MODE
```

```
U 17, 7700,8800,5BE4,0308,2470,1818  00FE   635 ;% 17   WCTRL/LOADCRAR,WB_R[TEMP2]                    ; ADDRESS OF S&C BIT IN CRAR
U 18, 7700,C800,5BE4,030C,2C70,1819  0109   639 ;% 18   WCTRL/CONWRITE,WB_R[TEMP3]                   ; DISABLES CTRL-P INTERRUPTS
U 19, 7300,C800,0364,0300,0470,181A  0114   643 ;% 19   STOP.CLOCK                                  ; HALT
                                     011F   647
U 1A, 7700,4800,0364,0300,0470,181B  011F   648 ;% 1A   NOP                                         ; GET THE CLOCK UP TO SPEED
U 1B, 7700,8800,5BE4,0310,2C70,181C  012A   652 ;% 1B   WCTRL/CONWRITE,WB_R[TEMP4]                   ; ENABLES CTRL-P INTERRUPTS
U 1C, 7700,C800,5BE4,03D8,0070,181D  0135   656 ;% 1C   WB_R[ZERO],CCPSL/PSL_WB.CCBR_ALUS          ; RESTORE CPU TO NATIVE MODE
U 1D, 7300,4800,0364,0300,0470,181E  0140   660 ;% 1D   STOP.CLOCK                                  ; HALT
                                     014B   664
                                     014B   665 RTEMP_DATA
                                     0001   666
                          00000000   0001   667         .LONG   0               ; HOLDS MODEL OPCODE
                          80000000   0005   668         .LONG   ^X80000000      ; DATA TO PLACE CPU INTO COMP. MODE
                          00C00000   0009   669         .LONG   ^X00C00000      ; ADDRESS OF STATUS AND CONTROL BIT IN CRAR
                          00400000   000D   670         .LONG   ^X00400000      ; DATA TO DISABLE CTRL-P INTERRUPTS
                          00900000   0011   671         .LONG   ^X00900000      ; DATA TO ENABLE CTRL-P INTERRUPTS
                                     0015   672
                                     0015   673 END_CPU_DATA
                                     00A4   674
                                     00A4   675 END_TEST_DATA
                                     00A4
                                     00A4       ;-------------------------------------------------------------------------
                                     00A4   676
                                     00A4   677 ENDTEST
```

```
0014          .SBTTL  "TEST   0D0     CM IR2 BUT Test
0014    679 ;*****************************************************************************
0014    680 ;++
0014    681 ;
0014    682 ; FUNCTIONAL DESCRIPTION
0014    683 ;
0014    684 ;       This test will examine the BUT/IR2 micro-branch the CPU is in
0014    685 ;       compatibility mode for proper operation.
0014    686 ;
0014    687 ; TEST ALGORITHM
0014    688 ;
0014    689 ;       Because the BUT/IR2 micro-branch actually uses CIR<2>, the opcode
0014    690 ;       bits used will be different depending on the class of the instruction
0014    691 ;       being decoded. To fully examine the micro-branch, this test will loop
0014    692 ;       until all the different opcodes have been tried.
0014    693 ;
0014    694 ;       The steps used are as follows:
0014    695 ;
0014    696 ;        1. Initialize the CPU.
0014    697 ;        2. Open the window to the signature analyzer.
0014    698 ;        3. Execute micro-code to place the CPU into compatibility mode and
0014    699 ;           to disable console Control-P interrupts.
0014    700 ;        4. Initialize the WD Register with a lonword of zero.
0014    701 ;        5. Loop for all IR2 decodes. (12 iterations)
0014    702 ;        6. Load an opcode from a table indexed by the loop count into WDREG.
0014    703 ;        7. Execute micro-code to move the opcode from the WDREG to RTEMP0.
0014    704 ;        8. Loop on error from here.
0014    705 ;        9. Execute the BUT/IR2 micro-branch, with 1800 in the NEXT field.
0014    706 ;       10. Do a masked comarison of the address in the Control Store address
0014    707 ;           TRaP register (CSTRP) with the expected address.
0014    708 ;       11. Execute micro-code to read back the contents of the IR and OSR.
0014    709 ;       12. If the expected and received control store addresses are not equal,
0014    710 ;           signal an error.
0014    711 ;       13. Otherwise continue the loop for all the opcodes.
0014    712 ;       14. Execute micro-code to enable console Control-P interrupts and to
0014    713 ;           return the CPU to native mode.
0014    714 ;       15. Close the window to the signature analyzer.
0014    715 ;       16. End of test.
0014    716 ;
0014    717 ; TEST DATA
0014    718 ;
0014    719 ;       Below is a table of the opcodes, expected control store addresses,
0014    720 ;       opcode values for CIR<2>, instruction mnemonic and class used for
0014    721 ;       each loop iteration of the test. Note that OPCODE bits are the same
0014    722 ;       as XB0 bits for this test.
0014    723 ;
0014    724 ;       Loop    Opcode  CS Address      CIR<2>          Instruction     Class
0014    725 ;
0014    726 ;         1.     0A00     1800         OPCODE<10>      CLR             A
0014    727 ;         2.     0C00     1801         OPCODE<10>      ROR             A
0014    728 ;         3.     3000     1800         OPCODE<14>      BIT             B1
0014    729 ;         4.     4000     1801         OPCODE<14>      BIC             B1
0014    730 ;         5.     7000     1800         OPCODE<10>      MUL             B2
0014    731 ;         6.     7400     1801         OPCODE<10>      XOR             B2
0014    732 ;         7.     0300     1800         OPCODE<10>      BEQ             C
0014    733 ;         8.     0400     1801         OPCODE<10>      BGE             C
0014    734 ;         9.     0000     1800         OPCODE<02>      HALT            D1
```

```
0014   ;        10.     0004     1801          OPCODE<02>      IOT             D1
0014   ;        11.     00A0     1800          OPCODE<04>      NOP             D2
0014   ;        12.     0098     1801          OPCODE<04>      SPL             D2
0014   ;
0014   ;       This table lists the initial values of the RTEMPS used in this test.
0014   ;
0014   ;       RTEMP0 = 0                Current opcode in the IR and OSR.
0014   ;       RTEMP1 = ^X80000000       Data to place CPU into compatibility mode.
0014   ;       RTEMP2 = ^X00C00000       Address of status and control bit in CRAR.
0014   ;       RTEMP3 = ^X00400000       Data to disable console Control-P interrupts.
0014   ;       RTEMP4 = ^X00900000       Data to enable console Control-P interrupts.
0014   ;
0014   ; LOGICAL DESCRIPTION
0014   ;
0014   ;       In this test the micro-branch is executed by driving CIR<2> onto bit 0
0014   ;       of the control store address. (CSA<0>) CSA<0> is determined by three
0014   ;       things; discrete logic on the DPM, the IRD gate array, and the PHB
0014   ;       gate array. CSA<0> is generated by an inverter that sits on the output
0014   ;       of an 8 to 1 multiplexer. The mux is enabled by the signal ENABLE BUT
0014   ;       4XXX, which is generated by a 1 of 8 decoder whose select lines depend
0014   ;       upon bits 5:3 of the BUT field. For the BUT/IR2 micro-order, input D2
0014   ;       is selected. This input comes directly from IR<2> on the IRD chip.
0014   ;       The selection of CIR<2> as opposed to IR<2> is determined by the IRD
0014   ;       gate array. IRD in turn uses the control signals IRD ADD CTL 1 and
0014   ;       IRD ADD CTL 0 from the PHB gate array.
0014   ;
0014   ; ASSUMPTIONS
0014   ;
0014   ;       This test assumes the following logic is working; the VA, PC, RTEMPS,
0014   ;       ALU, ROTATOR, the IR and OSR, and the instruction prefetch mechanism.
0014   ;
0014   ; ERROR DESCRIPTION
0014   ;
0014   ;       On error this test calls out the IRD and PHB gate arrays on the DPM.
0014   ;       The following information is also typed out:
0014   ;
0014   ;       EXPECTED CONTROL STORE ADDRESS
0014   ;       RECEVIED CONTROL STORE ADDRESS
0014   ;       LOOP COUNT
0014   ;       CURRENT CONTENTS OF THE IR AND OSR
0014   ;
0014   ;--
0014   ;*******************************************************************************
0014   ;///////////////////////////////////////////////////////////////////////////////
0014
0014           INITIALIZE                                  ; INITIALIZE THE CPU
0016           FETCH           <^X15>                      ; PLACE CPU INTO COMP. MODE AND
001B           BRSTCLK         <^X19>                      ; DISABLE CTRL-P INTERRUPTS
001F           LOADREG         WDREG,1$,,L                 ; ZERO THE WD REGISTER
0027           LOOP            I,1,12                      ; LOOP FOR ALL IR2 DECODES
002E           LOADREG         WDREG,2$,I,W                ; LOAD THE MODEL OPCODE
0036           FETCH           3                           ; LOAD OPCODE INTO IR AND OSR
003B           BRSTCLK         <^X0D>                      ; ...
003F           ERRLOOP                                     ; LOOP ON ERROR FROM HERE
0041           FETCH           0                           ; DO THE BUT/IR2 MICRO-BRANCH
0046           BRSTCLK         2                           ; ...
004A           CMPREGMSK       CSTRP,3$,I,4$,,W            ; WAS THE CS ADDRESS CORRECT?
```

```
                                0056   792          FETCH           <^X0E>                          ; WRITE IR AND OSR TO RTEMP0
                                005B   793          BRSTCLK         <^X14>,INH                      ; ...
                                005F   794          IFERROR         1,<<IRD><PHB>>                  ; IF ADDRESS BAD, SIGNAL ERROR
                                0066   795          ENDLOOP         I                               ; CONTINUE THE LOOP
                                0069   796          FETCH           <^X1A>                          ; RESTORE NATIVE MODE AND
                                006E   797          BRSTCLK         <^X1D>                          ; ENABLE CTRL-P INTERRUPTS
                                0072   798          SKIP                                            ; SKIP TO NEXT TEST
                                0079   799
                                0079   800 BEGIN_TEST_DATA
                                0079
                                0079       ;-------------------------------------------------------------------------
                                0079   801
                         00000000   0079   802 1$:      .LONG   0       ; LONGWORD TO ZERO WD REGISTER
                                007D   803 ;
                                007D   804 ; Test Opcodes
                                007D   805 ;
 0300 7400 7000 4000 3000 0C00 0A00   007D   806 2$:      .WORD   ^X0A00,^X0C00,^X3000,^X4000,^X7000,^X7400,^X0300
          0098 00A0 0004 0000 0400   008B   807           .WORD   ^X0400,^X0000,^X0004,^X00A0,^X0098
                                0095   808 ;
                                0095   809 ; Expected Control Store Addresses
                                0095   810 ;
 1800 1801 1800 1801 1800 1801 1800   0095   811 3$:      .WORD   ^X1800,^X1801,^X1800,^X1801,^X1800,^X1801,^X1800
          1801 1800 1801 1800 1801   00A3   812           .WORD   ^X1801,^X1800,^X1801,^X1800,^X1801
                                00AD   813
                             3FFF   00AD   814 4$:      .WORD   ^X3FFF  ; MASK FOR ADDRESS COMPARISON
                                00AF   815
                                00AF   816 BEGIN_CPU_DATA
                                0002   817
                                0002   818 DCS_DATA
                                0001   819
U 00, 7700,C800,0364,0300,0470,1801   0001   820 ;% 00   NOP                                         ; GET THE CLOCK UP TO SPEED
U 01, 7700,4800,0364,8B00,0470,1800   000C   824 ;% 01   NEXT/1800,BUT/IR2                           ; DO THE MICRO-BRANCH
U 02, 7300,4800,0364,0300,0470,1803   0017   828 ;% 02   STOP.CLOCK                                 ; HALT
                                0022   832
U 03, 7700,C800,0364,0300,0470,1804   0022   833 ;% 03   NOP                                         ; GET THE CLOCK UP TO SPEED
U 04, 5700,0840,0364,0300,0470,1805   002D   837 ;% 04   R[TEMP0]_RDM                               ; WRITE THE OPCODE TO RTEMP0
U 05, 7700,4800,5BE4,03D8,4A70,1806   0038   841 ;% 05   VA_R[ZERO]                                 ; ZERO THE VA
U 06, 7701,8800,5BE4,0200,5590,1807   0043   845 ;% 06   WRITE.LONG R[TEMP0]                        ; WRITE THE OPCODE TO CACHE
U 07, 7701,8800,5BE4,0200,55F0,1808   004E   849 ;% 07   WDR_UNROT(R[TEMP0]),BUS/PRB.RD,SIZE[LONG]
                                0059   853                                                         ; ...
U 08, 7700,C800,0364,0300,4470,1809   0059   854 ;% 08   VA_VA+4                                   ; INCREMENT THE VA
U 09, 7701,0800,5BE4,02D8,5590,180A   0064   858 ;% 09   WRITE.LONG R[ZERO]                         ; VALIDATE 2ND CACHE LOCATION
U 0A, 7701,8800,5BE4,02D8,55F0,180B   006F   862 ;% 0A   WDR_UNROT(R[ZERO]),BUS/PRB.RD,SIZE[LONG]
                                007A   866                                                         ; FOR XB1
U 0B, 7700,C800,5BE4,03D8,4870,180C   007A   867 ;% 0B   PC_R[ZERO]                                 ; INITIATE INSTRUCTION PREFETCH
U 0C, 7700,C800,0364,1700,0470,180D   0085   871 ;% 0C   BUT/IRD1TST                               ; LOAD THE IR AND OSR
U 0D, 7300,C800,0364,0300,0470,180E   0090   875 ;% 0D   STOP.CLOCK                                 ; HALT
                                009B   879
U 0E, 7700,4800,0364,0300,0470,180F   009B   880 ;% 0E   NOP                                         ; GET THE CLOCK UP TO SPEED
U 0F, 7700,4800,0364,0300,5670,1810   00A6   884 ;% 0F   WCTRL/MDR_IR                               ; WRITE THE IR TO THE MDR
U 10, 7700,D800,EF64,0304,0470,1811   00B1   888 ;% 10   PL_[8]                                     ; PUT AN 8 INTO THE P-LATCH
U 11, 7700,0852,8770,0300,0470,1812   00BC   892 ;% 11   R[TEMP0]_M[MDR].RL.P                       ; RTEMP0 GETS IR IN 2ND BYTE
U 12, 7700,C800,0364,0300,5E70,1813   00C7   896 ;% 12   CCPSL/MDR_OSR.CCBR_BRATST                 ; WRITE THE OSR TO THE MDR
U 13, 7700,4852,0024,0300,0470,1814   00D2   900 ;% 13   R[TEMP0]_M[MDR].OR.RB                     ; RTEMP0 = INSTRUCTION OPCODE
U 14, 7300,C800,0364,0300,0470,1815   00DD   904 ;% 14   STOP.CLOCK                                 ; HALT
                                00E8   908
U 15, 7700,C800,0364,0300,0470,1816   00E8   909 ;% 15   NOP                                         ; GET THE CLOCK UP TO SPEED
```

```
U 16. 7700.8800.5BE4.0304.0070.1817  00F3   913 ;% 16   WB_R[TEMP1],CCPSL/PSL_WB.CCBR_ALUS      ; PLACE THE CPU INTO COMP. MODE
U 17. 7700.8800.5BE4.0308.2470.1818  00FE   917 ;% 17   WCTRL/LOADCRAR,WB_R[TEMP2]              ; ADDRESS OF S&C BIT IN CRAR
U 18. 7700.C800.5BE4.030C.2C70.1819  0109   921 ;% 18   WCTRL/CONWRITE,WB_R[TEMP3]              ; DISABLE CTRL-P INTERRUPTS
U 19. 7300.C800.0364.0300.0470.181A  0114   925 ;% 19   STOP.CLOCK                             ; HALT
                                     011F   929
U 1A. 7700.4800.0364.0300.0470.181B  011F   930 ;% 1A   NOP                                   ; GET THE CLOCK UP TO SPEED
U 1B. 7700.8800.5BE4.0310.2C70.181C  012A   934 ;% 1B   WCTRL/CONWRITE,WB_R[TEMP4]              ; ENABLE CTRL-P INTERRUPTS
U 1C. 7700.C800.5BE4.03D8.0070.181D  0135   938 ;% 1C   WB_R[ZERO],CCPSL/PSL_WB.CCBR_ALUS      ; RESTORE NATIVE MODE
U 1D. 7300.4800.0364.0300.0470.181E  0140   942 ;% 1D   STOP.CLOCK                             ; HALT
                                     014B   946
                                     014B   947 RTEMP_DATA
                                     0001   948
                          00000000   0001   949         .LONG   0                ; MODEL OPCODE
                          80000000   0005   950         .LONG   ^X80000000       ; DATA TO PLACE CPU INTO COMP. MODE
                          00C00000   0009   951         .LONG   ^X00C00000       ; ADDRESS OF STATUS AND CONTROL BIT IN CRAR
                          00400000   000D   952         .LONG   ^X00400000       ; DISABLES CTRL-P INTERRUPTS
                          00900000   0011   953         .LONG   ^X00900000       ; ENABLES CTRL-P INTERRUPTS
                                     0015   954
                                     0015   955 END_CPU_DATA
                                     00AF   956
                                     00AF   957 END_TEST_DATA
                                     00AF
                                     00AF       ;-------------------------------------------------------------------------
                                     00AF   958
                                     00AF   959 ENDTEST
```

```
0014          .SBTTL  "TEST   0D1     CM IR5 BUT Test
0014    961 ;*****************************************************************************
0014    962 ;++
0014    963 ;
0014    964 ; FUNCTIONAL DESCRIPTION
0014    965 ;
0014    966 ;       This test examines a compatibility mode BUT/IR5 micro-branch.
0014    967 ;       For compatibility mode the BUT/IR5 micro-branch is executed on CIR<5>.
0014    968 ;       Since CIR<5> is a decode of the instruction in the IR its value will
0014    969 ;       depend upon the class of the instruction as defined in Chart 4.4.1A
0014    970 ;       of the micro-sequencer specification.
0014    971 ;
0014    972 ; TEST ALGORITHM
0014    973 ;
0014    974 ;       To obtain all the CIR decodes this test has been divided into
0014    975 ;       six subtests. Each subtest trys a different instruction class for
0014    976 ;       CIR<5>.
0014    977 ;
0014    978 ;       Subtest 1
0014    979 ;
0014    980 ;       This subtest examines the micro-branch for a Class A instruction.
0014    981 ;       According to Chart 4.4.1A, a Class A instruction decodes CIR<5> using
0014    982 ;       OPCODE<8>. Therefore, this subtest will loop for two iterations,
0014    983 ;       toggling OPCODE<8>. OPCODE<8> is the same as XB0<8>.
0014    984 ;
0014    985 ;       Subtests 2 - 6
0014    986 ;
0014    987 ;       These subtests examine the micro-branch for the remaining classes of
0014    988 ;       compatiblity mode instructions. Since these classes force CIR<5> to
0014    989 ;       either a 1 or a 0, only one pass is needed through each subtest.
0014    990 ;
0014    991 ;       Here are the steps used to implement the subtests. Note that Subtest 1
0014    992 ;       will loop twice between steps 5 and 11 in order to toggle OPCODE<8>.
0014    993 ;
0014    994 ;         1. Initialize the CPU.
0014    995 ;         2. Open the window to the signature analyzer.
0014    996 ;         3. Execute the micro-code to place the CPU into compatibility mode and
0014    997 ;            to disable console Control-P interrupts so that the micro-branch
0014    998 ;            will not trap.
0014    999 ;         4. Initialize the WD Register on the RDM with a longword of zeros.
0014   1000 ;         5. Load a test opcode into the WD register on the RDM.
0014   1001 ;         6. Execute micro-code to write the test opcode from the WD register to
0014   1002 ;            the IR and OSR in the CPU.
0014   1003 ;         7. Loop on error from here.
0014   1004 ;         8. Execute the BUT/IR5 micro-branch with 1800 in the NEXT field so
0014   1005 ;            that the RDM will allow the branch to finish.
0014   1006 ;         9. Do a masked comparison of the address generated by the micro-branch
0014   1007 ;            against the expected address.
0014   1008 ;        10. Execute the micro-code to read the IR and OSR into RTEMP0.
0014   1009 ;        11. If the micro-branch and expected addresses are different, signal an
0014   1010 ;            error and dump out RTEMP0.
0014   1011 ;
0014   1012 ; TEST DATA
0014   1013 ;
0014   1014 ;       This table gives the expected control store address, and instruction
0014   1015 ;       opcode, mnemonic and class for each subtest.
0014   1016 ;
```

```
0014  1017 ;       Subtest #       CS Address      Opcode          Instruction     Class
0014  1018 ;
0014  1019 ;       Subtest 1       1800,1801       0A00,0B00       CLR,NEG         A
0014  1020 ;       Subtest 2       1800            1000            MOV             B1
0014  1021 ;       Subtest 3       1801            7000            MUL             B2
0014  1022 ;       Subtest 4       1800            0100            BR              C
0014  1023 ;       Subtest 5       1800            0000            HALT            D1
0014  1024 ;       Subtest 6       1801            0080            RTS             D2
0014  1025 ;
0014  1026 ;       This table gives the initial value of the RTEMP registers used.
0014  1027 ;
0014  1028 ;       RTEMP0 = 0                  Current opcode in the IR and OSR.
0014  1029 ;       RTEMP1 = ^X80000000         Data to place CPU into compatibility mode.
0014  1030 ;       RTEMP2 = ^X00C00000         Address of status and control bit in CRAR.
0014  1031 ;       RTEMP3 = ^X00400000         Data to disable console Control-P interrupts.
0014  1032 ;       RTEMP4 = ^X00900000         Data to enable console Control-P interrupts.
0014  1033 ;
0014  1034 ; LOGICAL DESCRIPTION
0014  1035 ;
0014  1036 ;       In this test the micro-branch is executed by ORing CIR<5> in bit 0 of
0014  1037 ;       the control store address (CSA<0>). CSA<0> is determined by three
0014  1038 ;       things; discrete logic on the DPM, the IRD gate array, and the PHB
0014  1039 ;       gate array. CSA<0> is generated by an inverter that sits on the output
0014  1040 ;       of an 8 to 1 multiplexer. The mux is enabled by the signal ENABLE BUT
0014  1041 ;       4XXX, which is generated by a 1 of 8 decoder whose select lines depend
0014  1042 ;       upon bits 5:3 of the BUT field. For the BUT/IR5 micro-order, input D3
0014  1043 ;       is selected. This input comes directly from IR<5> on the IRD chip.
0014  1044 ;       The selection of CIR<5> as opposed to IR<5> is determined by the IRD
0014  1045 ;       gate array. IRD in turn uses the control signals IRD ADD CTL 1 and
0014  1046 ;       IRD ADD CTL 0 from the PHB gate array.
0014  1047 ;
0014  1048 ; ASSUMPTIONS
0014  1049 ;
0014  1050 ;       This test assumes the following logic is working; the VA, PC, RTEMPS,
0014  1051 ;       ALU, ROTATOR, and the IR and OSR.
0014  1052 ;
0014  1053 ; ERROR DESCRIPTION
0014  1054 ;
0014  1055 ;       One error this test calls out the IRD and PHB gate arrays on the DPM.
0014  1056 ;       The following information is also typed out:
0014  1057 ;
0014  1058 ;       Subtest 1
0014  1059 ;
0014  1060 ;       EXPECTED CONTROL STORE ADDRESS
0014  1061 ;       RECEIVED CONTROL STORE ADDRESS
0014  1062 ;       LOOP COUNT
0014  1063 ;       CURRENT CONTENTS OF THE IR AND OSR
0014  1064 ;
0014  1065 ;       Subtests 2 - 6
0014  1066 ;
0014  1067 ;       EXPECTED CONTROL STORE ADDRESS
0014  1068 ;       RECEVIED CONTROL STORE ADDRESS
0014  1069 ;       CURRENT CONTENTS OF THE IR AND OSR
0014  1070 ;
0014  1071 ;--
0014  1072 ;*****************************************************************************
0014  1073
```

```
                        0014  1074          SUBTEST         1
                        0017
                        0017          ;////////////////////////////////////////////////////////////////////////////////
                        0017  1075    ;+
                        0017  1076    ; According to Chart 4.4.1A, a Class A instruction decodes CIR<5> using OPCODE
                        0017  1077    ; <8>. Therefore, this subtest will loop for two iterations toggling OPCODE<8>.
                        0017  1078    ; For this subtest OPCODE<8> is the same as XB0<8>.
                        0017  1079    ;-
                        0017  1080          INITIALIZE                              ; INITIALIZE THE CPU
                        0019  1081          FETCH           <^X15>                  ; PLACE CPU INTO COMP. MODE AND
                        001E  1082          BRSTCLK         <^X19>                  ; DISABLE CTRL-P INTERRUPTS
                        0022  1083          LOADREG         WDREG,1$,,L             ; INITIALIZE THE WD REGISTER
                        002A  1084          LOOP            I,1,2                   ; LOOP FOR BOTH CLASS A OPCODES
                        0031  1085          LOADREG         WDREG,2$,I,W            ; LOAD MODEL OPCODE INTO WDREG
                        0039  1086          FETCH           3                       ; LOAD OPCODE INTO IR AND OSR
                        003E  1087          BRSTCLK         <^X0D>                  ; ...
                        0042  1088          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        0044  1089          FETCH           0                       ; EXECUTE BUT/IR5 MICRO-BRANCH
                        0049  1090          BRSTCLK         2                       ; ...
                        004D  1091          CMPREGMSK       CSTRP,8$,I,10$,,W       ; IS THE ADDRESS CORRECT?
                        0059  1092          FETCH           <^X0E>                  ; READ BACK THE IR AND OSR
                        005E  1093          BRSTCLK         <^X14>,INH              ; ...
                        0062  1094          IFERROR         1,<<IRD><PHB>>          ; IF BAD ADDRESS SIGNAL ERROR
                        0069  1095          ENDLOOP         I                       ; CONTINUE LOOP FOR OPCODES
                        006C  1096
                        006C  1097          SUBTEST         2
                        006F
                        006F          ;////////////////////////////////////////////////////////////////////////////////
                        006F  1098    ;+
                        006F  1099    ; This subtest examines the CIR<5> micro-branch for a Class B1 instruction.
                        006F  1100    ; A Class B1 instruction forces CIR<5> to 0.
                        006F  1101    ;-
                        006F  1102          LOADREG         WDREG,3$,,W             ; LOAD MODEL OPCODE INTO WDREG
                        0077  1103          FETCH           3                       ; LOAD OPCODE INTO IR AND OSR
                        007C  1104          BRSTCLK         <^X0D>                  ; ...
                        0080  1105          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        0082  1106          FETCH           0                       ; EXECUTE BUT/IR5 MICRO-BRANCH
                        0087  1107          BRSTCLK         2                       ; ...
                        008B  1108          CMPREGMSK       CSTRP,8$,,10$,,W        ; IS THE ADDRESS CORRECT?
                        0097  1109          FETCH           <^X0E>                  ; READ BACK THE IR AND OSR
                        009C  1110          BRSTCLK         <^X14>,INH              ; ...
                        00A0  1111          IFERROR         1,<<IRD><PHB>>          ; IF BAD ADDRESS SIGNAL ERROR
                        00A7  1112
                        00A7  1113          SUBTEST         3
                        00AA
                        00AA          ;////////////////////////////////////////////////////////////////////////////////
                        00AA  1114    ;+
                        00AA  1115    ; This subtest examines the CIR<5> micro-branch for a Class B2 instruction.
                        00AA  1116    ; A Class B2 instruction forces CIR<5> to 1.
                        00AA  1117    ;-
                        00AA  1118          LOADREG         WDREG,4$,,W             ; LOAD MODEL OPCODE INTO WDREG
                        00B2  1119          FETCH           3                       ; LOAD OPCODE INTO IR AND OSR
                        00B7  1120          BRSTCLK         <^X0D>                  ; ...
                        00BB  1121          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        00BD  1122          FETCH           0                       ; EXECUTE BUT/IR5 MICRO-BRANCH
                        00C2  1123          BRSTCLK         2                       ; ...
                        00C6  1124          CMPREGMSK       CSTRP,9$,,10$,,W        ; IS THE ADDRESS CORRECT?
```

```
00D2  1125          FETCH           <^X0E>                  ; READ BACK THE IR AND OSR
00D7  1126          BRSTCLK         <^X14>,INH              ; ...
00DB  1127          IFERROR         1,<<IRD><PHB>>          ; IF BAD ADDRESS SIGNAL ERROR
00E2  1128
00E2  1129          SUBTEST         4
00E5
00E5          ;////////////////////////////////////////////////////////////////////////////////
00E5  1130 ;+
00E5  1131 ; This subtest examines the CIR<5> micro-branch for a Class C instruction.
00E5  1132 ; A Class C instruction forces CIR<5> to 0.
00E5  1133 ;-
00E5  1134          LOADREG         WDREG,5$,,W             ; LOAD MODEL OPCODE INTO WDREG
00ED  1135          FETCH           3                       ; LOAD OPCODE INTO IR AND OSR
00F2  1136          BRSTCLK         <^X0D>                  ; ...
00F6  1137          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
00F8  1138          FETCH           0                       ; EXECUTE BUT/IR5 MICRO-BRANCH
00FD  1139          BRSTCLK         2                       ; ...
0101  1140          CMPREGMSK       CSTRP,8$,,10$,,W        ; IS THE ADDRESS CORRECT?
010D  1141          FETCH           <^X0E>                  ; READ BACK THE IR AND OSR
0112  1142          BRSTCLK         <^X14>,INH              ; ...
0116  1143          IFERROR         1,<<IRD><PHB>>          ; IF BAD ADDRESS SIGNAL ERROR
011D  1144
011D  1145          SUBTEST         5
0120
0120          ;////////////////////////////////////////////////////////////////////////////////
0120  1146 ;+
0120  1147 ; This subtest examines the CIR<5> micro-branch for a Class D1 instruction.
0120  1148 ; A Class D1 instruction forces CIR<5> to 0.
0120  1149 ;-
0120  1150          LOADREG         WDREG,6$,,W             ; LOAD MODEL OPCODE INTO WDREG
0128  1151          FETCH           3                       ; LOAD OPCODE INTO IR AND OSR
012D  1152          BRSTCLK         <^X0D>                  ; ...
0131  1153          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
0133  1154          FETCH           0                       ; EXECUTE BUT/IR5 MICRO-BRANCH
0138  1155          BRSTCLK         2                       ; ...
013C  1156          CMPREGMSK       CSTRP,8$,,10$,,W        ; IS THE ADDRESS CORRECT?
0148  1157          FETCH           <^X0E>                  ; READ BACK THE IR AND OSR
014D  1158          BRSTCLK         <^X14>,INH              ; ...
0151  1159          IFERROR         1,<<IRD><PHB>>          ; IF BAD ADDRESS SIGNAL ERROR
0158  1160
0158  1161          SUBTEST         6
015B
015B          ;////////////////////////////////////////////////////////////////////////////////
015B  1162 ;+
015B  1163 ; This subtest examines the CIR<5> micro-branch for a Class D2 instruction.
015B  1164 ; A Class D2 instruction forces CIR<5> to 1.
015B  1165 ;-
015B  1166          LOADREG         WDREG,7$,,W             ; LOAD MODEL OPCODE INTO WDREG
0163  1167          FETCH           3                       ; LOAD OPCODE INTO IR AND OSR
0168  1168          BRSTCLK         <^X0D>                  ; ...
016C  1169          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
016E  1170          FETCH           0                       ; EXECUTE BUT/IR5 MICRO-BRANCH
0173  1171          BRSTCLK         2                       ; ...
0177  1172          CMPREGMSK       CSTRP,9$,,10$,,W        ; IS THE ADDRESS CORRECT?
0183  1173          FETCH           <^X0E>                  ; READ BACK THE IR AND OSR
0188  1174          BRSTCLK         <^X14>,INH              ; ...
018C  1175          IFERROR         1,<<IRD><PHB>>          ; IF BAD ADDRESS SIGNAL ERROR
```

```
                        0193  1176
                        0193  1177          FETCH           <^X1A>                  ; PUT CPU BACK INTO NATIVE MODE
                        0198  1178          BRSTCLK         <^X1D>                  ; AND ENABLE CTRL-P INTERRUPTS
                        019C  1179          SKIP                                    ; END OF TEST
                        01A3  1180
                        01A3  1181 BEGIN_TEST_DATA
                        01A3
                        01A3       ;-------------------------------------------------------------------------
                        01A3  1182
 00000000               01A3  1183 1$:      .LONG   0       ; LONGWORD TO INTIALIZE THE WD REGISTER
                        01A7  1184 ;
                        01A7  1185 ; Test Opcodes
                        01A7  1186 ;
 0B00 0A00              01A7  1187 2$:      .WORD   ^X0A00,^X0B00   ; CLR, NEG
      1000              01AB  1188 3$:      .WORD   ^X1000          ; MOV
      7000              01AD  1189 4$:      .WORD   ^X7000          ; MUL
      0100              01AF  1190 5$:      .WORD   ^X0100          ; BR
      0000              01B1  1191 6$:      .WORD   ^X0000          ; HALT
      0080              01B3  1192 7$:      .WORD   ^X0080          ; RTS
                        01B5  1193 ;
                        01B5  1194 ; Expected Control Store Addresses
                        01B5  1195 ;
      1800              01B5  1196 8$:      .WORD   ^X1800
      1801              01B7  1197 9$:      .WORD   ^X1801
                        01B9  1198
      3FFF              01B9  1199 10$:     .WORD   ^X3FFF  ; MASK FOR ADDRESS COMPARISON
                        01BB  1200
                        01BB  1201 BEGIN_CPU_DATA
                        0002  1202
                        0002  1203 DCS_DATA
                        0001  1204
U 00, 7700,C800,0364,0300,0470,1801     0001  1205 ;% 00    NOP                                     ; GET CLOCK UP TO SPEED
U 01, 7700,0800,0364,8F00,0470,1800     000C  1209 ;% 01    NEXT/1800,BUT/IR5                       ; EXECUTE THE MICRO-BRANCH
U 02, 7300,4800,0364,0300,0470,1803     0017  1213 ;% 02    STOP.CLOCK                              ; HALT
                                        0022  1217
U 03, 7700,C800,0364,0300,0470,1804     0022  1218 ;% 03    NOP                                     ; GET CLOCK UP TO SPEED
U 04, 5700,0840,0364,0300,0470,1805     002D  1222 ;% 04    R[TEMP0]_RDM                            ; WRITE OPCODES TO RTEMP0
U 05, 7700,4800,5BE4,03D8,4A70,1806     0038  1226 ;% 05    VA_R[ZERO]                              ; ZERO THE VA
U 06, 7701,8800,5BE4,0200,5590,1807     0043  1230 ;% 06    WRITE.LONG R[TEMP0]                     ; WRITE OPCODE TO CACHE
U 07, 7701,8800,5BE4,0200,55F0,1808     004E  1234 ;% 07    WDR_UNROT(R[TEMP0]),BUS/PRB.RD,SIZE[LONG]
                                        0059  1238                                              ;
U 08, 7700,C800,0364,0300,4470,1809     0059  1239 ;% 08    VA_VA+4                                 ; INCREMENT THE VA
U 09, 7701,0800,5BE4,02D8,5590,180A     0064  1243 ;% 09    WRITE.LONG R[ZERO]                      ; WRITE 2ND LOCATION FOR XB1
U 0A, 7701,8800,5BE4,02D8,55F0,180B     006F  1247 ;% 0A    WDR_UNROT(R[ZERO]),BUS/PRB.RD,SIZE[LONG]
                                        007A  1251                                              ;
U 0B, 7700,C800,5BE4,03D8,4870,180C     007A  1252 ;% 0B    PC_R[ZERO]                              ; INITIATE INSTRUCTION PREFETCH
U 0C, 7700,C800,0364,1700,0470,180D     0085  1256 ;% 0C    BUT/IRD1TST                             ; LOAD THE IR AND OSR
U 0D, 7300,C800,0364,0300,0470,180E     0090  1260 ;% 0D    STOP.CLOCK                              ; HALT
                                        009B  1264
U 0E, 7700,4800,0364,0300,0470,180F     009B  1265 ;% 0E    NOP                                     ; GET THE CLOCK UP TO SPEED
U 0F, 7700,4800,0364,0300,5670,1810     00A6  1269 ;% 0F    WCTRL/MDR_IR                            ; WRITE THE IR INTO THE MDR
U 10, 7700,D800,EF64,0304,0470,1811     00B1  1273 ;% 10    PL_[8]                                  ; PUT AN 8 IN THE P-LATCH
U 11, 7700,0852,8770,0300,0470,1812     00BC  1277 ;% 11    R[TEMP0]_M[MDR].RL.P                    ; RTEMP0 = IR .RL. 8 BITS
U 12, 7700,C800,0364,0300,5E70,1813     00C7  1281 ;% 12    CCPSL/MDR_OSR.CCBR_BRATST               ; WRITE THE OSR INTO THE MDR
U 13, 7700,4852,0024,0300,0470,1814     00D2  1285 ;% 13    R[TEMP0]_M[MDR].OR.RB                   ; RTEMP0 = INSTRUCTION OPCODE
U 14, 7300,C800,0364,0300,0470,1815     00DD  1289 ;% 14    STOP.CLOCK                              ; HALT
                                        00E8  1293
```

```
U 15, 7700,C800,0364,0300,0470,1816  00E8  1294 ;% 15   NOP                                          ; GET THE CLOCK UP TO SPEED
U 16, 7700,8800,5BE4,0304,0070,1817  00F3  1298 ;% 16   WB_R[TEMP1],CCPSL/PSL_WB.CCBR_ALUS          ; PLACE CPU INTO COMP. MODE
U 17, 7700,8800,5BE4,0308,2470,1818  00FE  1302 ;% 17   WCTRL/LOADCRAR,WB_R[TEMP2]                  ; ADDRESS OF S&C BIT IN CRAR
U 18, 7700,C800,5BE4,030C,2C70,1819  0109  1306 ;% 18   WCTRL/CONWRITE,WB_R[TEMP3]                  ; DISABLES CTRL-P INTERRUPTS
U 19, 7300,C800,0364,0300,0470,181A  0114  1310 ;% 19   STOP.CLOCK                                  ; HALT
                                     011F  1314
U 1A, 7700,4800,0364,0300,0470,181B  011F  1315 ;% 1A   NOP                                          ; GET CLOCK UP TO SPEED
U 1B, 7700,8800,5BE4,0310,2C70,181C  012A  1319 ;% 1B   WCTRL/CONWRITE,WB_R[TEMP4]                  ; ENABLE CTRL-P INTERRUPTS
U 1C, 7700,C800,5BE4,03D8,0070,181D  0135  1323 ;% 1C   WB_R[ZERO],CCPSL/PSL_WB.CCBR_ALUS          ; RESTORES CPU TO NATIVE MODE
U 1D, 7300,4800,0364,0300,0470,181E  0140  1327 ;% 1D   STOP.CLOCK                                  ; HALT
                                     014B  1331
                                     014B  1332 RTEMP_DATA
                                     0001  1333
                         00000000    0001  1334          .LONG   0                ; HOLDS MODEL OPCODE
                         80000000    0005  1335          .LONG   ^X80000000       ; PLACES CPU INTO COMP. MODE
                         00C00000    0009  1336          .LONG   ^X00C00000       ; ADDRESS OF STATUS AND CONTROL BIT IN CRAR
                         00400000    000D  1337          .LONG   ^X00400000       ; DISABLES CTRL-P INTERRUPTS
                         00900000    0011  1338          .LONG   ^X00900000       ; ENABLES CTRL-P INTERRUPTS
                                     0015  1339
                                     0015  1340 END_CPU_DATA
                                     01BB  1341
                                     01BB  1342 END_TEST_DATA
                                     01BB
                                     01BB       ;-------------------------------------------------------------------------
                                     01BB  1343
                                     01BB  1344 ENDTEST
```

```
                        001E        .SBTTL  "TEST   0D2     CM Branch on Address Test
                        001E  1346 ;*******************************************************************************
                        001E  1347 ;++
                        001E  1348 ;
                        001E  1349 ; FUNCTIONAL DESCRIPTION
                        001E  1350 ;
                        001E  1351 ;       This test loads the IR and OSR with model opcodes. It then executes a
                        001E  1352 ;       BUT/BRA.ON.ADD to see if the address that is generated is equal to the
                        001E  1353 ;       correct offset for that mode.
                        001E  1354 ;
                        001E  1355 ; TEST ALGORITHM
                        001E  1356 ;
                        001E  1357 ;         1. Initialize the CPU.
                        001E  1358 ;         2. Open the window for the signature analyzer.
                        001E  1359 ;         3. Execute micro-code to:
                        001E  1360 ;                 1. Set the compatibility mode bit in the PSL.
                        001E  1361 ;                 2. Disable a console CTRL-P interrupt after a BUT micro-order.
                        001E  1362 ;         4. Initialize the WD register with zero.
                        001E  1363 ;         5. Set up to expect non-DCS addresses.
                        001E  1364 ;         6. Prepare a loop for all possible offsets. (17 iterations)
                        001E  1365 ;         7. Load the test opcode into the RDM WD register.
                        001E  1366 ;         8. Execute the following micro-code to set up the test:
                        001E  1367 ;                 3. Load the model opcode from the WDREG to RTEMP0.
                        001E  1368 ;                 4. Write RTEMP0 to cache at physical location 0.
                        001E  1369 ;                 5. Increment the VA by 4.
                        001E  1370 ;                 6. Write a longword a zeroes to cache to validate it for
                        001E  1371 ;                    instruction prefetch.
                        001E  1372 ;                 7. Load the PC with zero to initiate instruction prefetch.
                        001E  1373 ;                 8. Execute a BUT/IRD1TST to load the IR and OSR.
                        001E  1374 ;         9. Loop on error from this point.
                        001E  1375 ;        10. Execute the following test micro-code:
                        001E  1376 ;                 9. NEXT/1800,BUT/BRA.ON.ADD
                        001E  1377 ;        11. Do a masked comparison of the address in the Control Store Address
                        001E  1378 ;            Trap register with the expected control store address.
                        001E  1379 ;        12. Execute micro-code to:
                        001E  1380 ;                10. Place the zero extended contents of the IR into the MDR.
                        001E  1381 ;                11. Rotate the MDR left by 8 bits and put it into RTEMP0.
                        001E  1382 ;                12. Place the zero extended contents of the OSR into the MDR.
                        001E  1383 ;                13. Or the MDR with RTEMP0 and put the results into RTEMP0.
                        001E  1384 ;        13. If the expected and received control store addresses were different
                        001E  1385 ;            signal an error and dump out the contents of the IR and OSR that
                        001E  1386 ;            were placed in RTEMP0.
                        001E  1387 ;        14. Continue the loop started in step 6.
                        001E  1388 ;        15. Execute micro-code to:
                        001E  1389 ;                14. Enable CTRL-P interrupts after a BUT micro-order.
                        001E  1390 ;                15. Zero out the PSL to restore native mode.
                        001E  1391 ;        16. Close the window to the signature analyzer.
                        001E  1392 ;        17. End of test.
                        001E  1393 ;
                        001E  1394 ; TEST DATA
                        001E  1395 ;
                        001E  1396 ;       This test uses the compatibility mode MOV instruction with all possible
                        001E  1397 ;       register and program counter addressing modes. The CS addresses
                        001E  1398 ;       generated will be within DCS range to allow the test micro-branch to
                        001E  1399 ;       complete execution.
                        001E  1400 ;
                        001E  1401 ;       Below is a table that shows the different opcodes, expected control
```

```
001E  1402 ;       store addresses, operands and the associated modes used for each loop.
001E  1403 ;
001E  1404 ;       Loop    Opcode  CS Address      Operands         Mode
001E  1405 ;
001E  1406 ;         1.      1000      1800        R0,R0            REGISTER
001E  1407 ;         2.      1007      1801        R0,R7            REGISTER MODE PC
001E  1408 ;         3.      1008      1802        R0,(R0)          REGISTER DEFERRED
001E  1409 ;         4.      100F      1803        R0,(R7)          REGISTER DEFERRED PC
001E  1410 ;         5.      1010      1804        R0,(R0)+         AUTO-INCR
001E  1411 ;         6.      1016      1805        R0,(R6)+         AUTO-INCR STACK POINTER
001E  1412 ;         7.      1018      1806        R0,@(R0)+        AUTO-INCR DEFERRED
001E  1413 ;         8.      101F      1807        R0,@#Addr        AUTO-INCR DEFERRED PC
001E  1414 ;         9.      1020      1808        R0,-(R0)         AUTO-DECR
001E  1415 ;         A.      1026      1809        R0,-(R6)         AUTO-DECR STACK POINTER
001E  1416 ;         B.      1028      180A        R0,@-(R0)        AUTO-DECR DEFERRED
001E  1417 ;         C.      1027      180B        R0,-(R7)         AUTO-DECR PC
001E  1418 ;         D.      1030      180C        R0,X(R0)         INDEX
001E  1419 ;         E.      1037      180D        R0,Addr X(R7)    RELATIVE
001E  1420 ;         F.      1038      180E        R0,@Addr @X(R0)  INDEX DEFERRED
001E  1421 ;        10.      103F      180E        R0,@Addr @X(R7)  RELATIVE DEFERRED
001E  1422 ;        11.      1017      180F        R0,#             IMMEDIATE
001E  1423 ;
001E  1424 ; LOGICAL DESCRIPTION
001E  1425 ;
001E  1426 ;       The logic involved in the branch on address micro-order is contained
001E  1427 ;       inside the IRD and PHB gate arrays. For this test the control store
001E  1428 ;       addresses generated will depend on the second operand specifier which
001E  1429 ;       has been loaded into the OSR, the micro-order itself, and the control
001E  1430 ;       signals IRD ADD CTL1, IRD ADD CTL 0 and PSL CM which come from the PHB
001E  1431 ;       gate array.
001E  1432 ;
001E  1433 ; ASSUMPTIONS
001E  1434 ;
001E  1435 ;       This test assumes the following logic has been tested prior to the
001E  1436 ;       execution of this test. The RTEMP registers, and the IR and OSR.
001E  1437 ;
001E  1438 ; ERROR DESCRIPTION
001E  1439 ;
001E  1440 ;       On error this test will call out the IRD and PHB gate arrays and
001E  1441 ;       the following information:
001E  1442 ;
001E  1443 ;       EXPECTED CONTROL STORE ADDRESS
001E  1444 ;       RECEIVED CONTROL STORE ADDRESS
001E  1445 ;       LOOP COUNT
001E  1446 ;       OPCODE OF THE INSTRUCTION BEING DECODED
001E  1447 ;
001E  1448 ;--
001E  1449 ;*******************************************************************************
001E  1450 ;///////////////////////////////////////////////////////////////////////////////
001E  1451
001E  1452         INITIALIZE                                  ; INITIALIZE THE CPU
0020  1453         FETCH           <^X15>                      ; PLACES CPU INTO COMP. MODE
0025  1454         BRSTCLK         <^X19>                      ; ...
0029  1455         LOADREG         WDREG,1$,,L                 ; INITIALIZE THE WD REGISTER
0031  1456         EXPUTRAP                                    ; EXPECT NON-DCS ADDRESSES
0033  1457         LOOP            I,1,17                      ; LOOP FOR 17 PATTERNS
003A  1458         LOADREG         WDREG,2$,I,W                ; LOAD THE TEST OPCODE
```

```
                                0042  1459          FETCH           <^X03>                  ; SET UP FOR THE TEST
                                0047  1460          BRSTCLK         <^X0D>                  ; ...
                                004B  1461          ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                                004D  1462          FETCH           0                       ; EXECUTE THE TEST
                                0052  1463          BRSTCLK         2                       ; ...
                                0056  1464          CMPREGMSK       CSTRP,3$,I,4$,,W        ; IS THE ADDRESS CORRECT?
                                0062  1465          FETCH           <^X0E>                  ; FETCH THE MODEL OPCODE
                                0067  1466          BRSTCLK         <^X14>,INH              ; FROM THE IR AND OSR
                                006B  1467          IFERROR         1,<<IRD><PHB>>          ; IF BAD ADDRESS, SIGNAL ERROR
                                0072  1468          ENDLOOP         I                       ; CONTINUE LOOP
                                0075  1469          FETCH           <^X1A>                  ; RESTORE NATIVE MODE
                                007A  1470          BRSTCLK         <^X1D>                  ; ...
                                007E  1471          SKIP                                    ; END OF TEST
                                0085  1472
                                0085  1473 BEGIN_TEST_DATA
                                0085
                                0085        ;-----------------------------------------------------------------------
                                0085  1474
                    00000000    0085  1475 1$:      .LONG   0
                                0089  1476
1018 1016 1010 100F 1008 1007 1000  0089  1477 2$:      .WORD   ^X1000,^X1007,^X1008,^X100F,^X1010,^X1016,^X1018 ; 01 - 07
1037 1030 1027 1028 1026 1020 101F  0097  1478          .WORD   ^X101F,^X1020,^X1026,^X1028,^X1027,^X1030,^X1037 ; 08 - 0E
                    1017 103F 1038  00A5  1479          .WORD   ^X1038,^X103F,^X1017                             ; 0F - 11
                                00AB  1480
1806 1805 1804 1803 1802 1801 1800  00AB  1481 3$:      .WORD   ^X1800,^X1801,^X1802,^X1803,^X1804,^X1805,^X1806 ; 01 - 07
180D 180C 180B 180A 1809 1808 1807  00B9  1482          .WORD   ^X1807,^X1808,^X1809,^X180A,^X180B,^X180C,^X180D ; 08 - 0E
                    180F 180E 180E  00C7  1483          .WORD   ^X180E,^X180E,^X180F                             ; 0F - 11
                                00CD  1484
                          3FFF  00CD  1485 4$:      .WORD   ^X3FFF  ; MASK FOR ADDRESS COMPARISON
                                00CF  1486
                                00CF  1487 BEGIN_CPU_DATA
                                0002  1488
                                0002  1489 DCS_DATA
                                0001  1490
U 00, 7700,C800,0364,0300,0470,1801 0001  1491 ;% 00   NOP                                     ; ALLOW CLOCK TO GET UP TO SPEED
U 01, 7700,4800,0364,6300,0470,1800 000C  1495 ;% 01   NEXT/1800,BUT/BRA.ON.ADD              ; EXECUTE THE BUT/BRANCH.ON.ADDRESS
U 02, 7300,4800,0364,0300,0470,1803 0017  1499 ;% 02   STOP.CLOCK                           ; HALT
                                0022  1503
U 03, 7700,C800,0364,0300,0470,1804 0022  1504 ;% 03   NOP                                     ; GET CLOCK UP TO SPEED
U 04, 5700,0840,0364,0300,0470,1805 002D  1508 ;% 04   R[TEMP0]_RDM                          ; LOAD THE MODEL OPCODE INTO RTEMP0
U 05, 7700,4800,5BE4,03D8,4A70,1806 0038  1512 ;% 05   VA_R[ZERO]                           ; ZERO THE VA
U 06, 7701,8800,5BE4,0200,5590,1807 0043  1516 ;% 06   WRITE.LONG R[TEMP0]                  ; WRITE THE MODEL OPCODE TO CACHE
U 07, 7701,8800,5BE4,0200,55F0,1808 004E  1520 ;% 07   WDR_UNROT(R[TEMP0]),BUS/PRB.RD,SIZE[LONG]
                                0059  1524                                              ; HOLD FOR SECOND CYCLE DUE TO CMI OFF
U 08, 7700,C800,0364,0300,4470,1809 0059  1525 ;% 08   VA_VA+4                             ; INCREMENT THE VA
U 09, 7701,0800,5BE4,02D8,5590,180A 0064  1529 ;% 09   WRITE.LONG R[ZERO]                  ; WRITE ZERO TO CACHE TO VALIDATE
U 0A, 7701,8800,5BE4,02D8,55F0,180B 006F  1533 ;% 0A   WDR_UNROT(R[ZERO]),BUS/PRB.RD,SIZE[LONG]
                                007A  1537                                              ; SECOND LOCATION FOR XB1
U 0B, 7700,C800,5BE4,03D8,4870,180C 007A  1538 ;% 0B   PC_R[ZERO]                           ; ZERO PC TO START INSTRUCTION PREFETCH
U 0C, 7700,C800,0364,1700,0470,180D 0085  1542 ;% 0C   BUT/IRD1TST                          ; LOAD THE IR AND OSR
U 0D, 7300,C800,0364,0300,0470,180E 0090  1546 ;% 0D   STOP.CLOCK                           ; HALT
                                009B  1550
U 0E, 7700,4800,0364,0300,0470,180F 009B  1551 ;% 0E   NOP                                     ; GET CLOCK UP TO SPEED
U 0F, 7700,4800,0364,0300,5670,1810 00A6  1555 ;% 0F   WCTRL/MDR_IR                          ; READ IR INTO MDR ZERO EXTENDED
U 10, 7700,D800,EF64,0304,0470,1811 00B1  1559 ;% 10   PL_[8]                               ; PUT AN 8 INTO THE P LATCH
U 11, 7700,0852,8770,0300,0470,1812 00BC  1563 ;% 11   R[TEMP0]_M[MDR].RL.P                 ; SHIFT MDR LEFT 8 BITS WRITE TO RTEMP0
U 12, 7700,C800,0364,0300,5E70,1813 00C7  1567 ;% 12   CCPSL/MDR_OSR.CCBR_BRATST           ; READ OSR INTO MDR ZERO EXTENDED
```

```
U 13. 7700.4852.0024.0300.0470.1814  00D2  1571 ;% 13   R[TEMP0]_M[MDR].OR.RB                ; OR MDR WITH RTEMP0 WRITE TO RTEMP0
U 14. 7300.C800.0364.0300.0470.1815  00DD  1575 ;% 14   STOP.CLOCK                           ; HALT
                                     00E8  1579
U 15. 7700.C800.0364.0300.0470.1816  00E8  1580 ;% 15   NOP                                 ; GET CLOCK UP TO SPEED
U 16. 7700.8800.5BE4.0304.0070.1817  00F3  1584 ;% 16   WB_R[TEMP1],CCPSL/PSL_WB.CCBR_ALUS ; PUT CPU INTO COMPATIBILITY MODE
U 17. 7700.8800.5BE4.0308.2470.1818  00FE  1588 ;% 17   WCTRL/LOADCRAR,WB_R[TEMP2]          ; SET STATUS AND CONTROL BIT IN CRAR
U 18. 7700.C800.5BE4.030C.2C70.1819  0109  1592 ;% 18   WCTRL/CONWRITE,WB_R[TEMP3]         ; DISABLE CTRL-P INTERRUPTS
U 19. 7300.C800.0364.0300.0470.181A  0114  1596 ;% 19   STOP.CLOCK                           ; HALT
                                     011F  1600
U 1A. 7700.4800.0364.0300.0470.181B  011F  1601 ;% 1A   NOP                                 ; GET CLOCK UP TO SPEED
U 1B. 7700.8800.5BE4.0310.2C70.181C  012A  1605 ;% 1B   WCTRL/CONWRITE,WB_R[TEMP4]         ; ENABLE CTRL-P INTERRUPTS
U 1C. 7700.C800.5BE4.03D8.0070.181D  0135  1609 ;% 1C   WB_R[ZERO],CCPSL/PSL_WB.CCBR_ALUS ; PUT CPU INTO NATIVE MODE
U 1D. 7300.4800.0364.0300.0470.181E  0140  1613 ;% 1D   STOP.CLOCK                           ; HALT
                                     014B  1617
                                     014B  1618 RTEMP_DATA
                                     0001  1619
                           00000000  0001  1620          .LONG   0                  ; HOLDS MODEL OPCODE
                           80000000  0005  1621          .LONG   ^X80000000         ; DATA TO PLACE CPU INTO COMP. MODE
                           00C00000  0009  1622          .LONG   ^X00C00000         ; ADDRESS OF STATUS AND CONTROL BIT IN CRAR
                           00400000  000D  1623          .LONG   ^X00400000         ; DATA TO DISABLE CTRL-P INTERRUPTS
                           00900000  0011  1624          .LONG   ^X00900000         ; DATA TO ENABLE CTRL-P INTERRUPTS
                                     0015  1625
                                     0015  1626 END_CPU_DATA
                                     00CF  1627
                                     00CF  1628 END_TEST_DATA
                                     00CF
                                     00CF          ;-------------------------------------------------------------------------
                                     00CF  1629
                                     00CF  1630 ENDTEST
```

```
0016        .SBTTL  "TEST   0D3     CM OS Branch Test
0016  1632 ;*******************************************************************************
0016  1633 ;++
0016  1634 ;
0016  1635 ; FUNCTIONAL DESCRIPTION
0016  1636 ;
0016  1637 ;       This test examines all of the mode bits in a compatibility mode operand
0016  1638 ;       specifier for all possible control store address offsets. The addresses
0016  1639 ;       generated are checked against chart 4.1.3B in the micro-sequencer spec.
0016  1640 ;
0016  1641 ; TEST ALGORITHM
0016  1642 ;
0016  1643 ;       The algorithm used to implement this test is as follows:
0016  1644 ;
0016  1645 ;         1. Initialize the CPU.
0016  1646 ;         2. Open the window to the signature analyzer.
0016  1647 ;         3. Execute the micro-code to put the CPU into compatibility mode.
0016  1648 ;         4. Prepare the RDM to expect non-DCS addresses.
0016  1649 ;         5. Initialize the WD register with zero.
0016  1650 ;         6. Prepare a loop for all operand modes. (17 iterations)
0016  1651 ;         7. Load a model opcode into the WD register.
0016  1652 ;         8. Loop on error from here.
0016  1653 ;         9. Execute the test micro-code:
0016  1654 ;                 1. Write the model opcode from WDREG to RTEMP0.
0016  1655 ;                 2. Zero the VA.
0016  1656 ;                 3. Write the model opcode to cache.
0016  1657 ;                 4. Increment the VA.
0016  1658 ;                 5. Write zero to cache to validate location for XB1.
0016  1659 ;                 6. Zero the PC to initiate instruction prefetch.
0016  1660 ;                 7. Execute a BUT/IRD1 to decode the instruction.
0016  1661 ;        10. Mask the address in the Control Store address TRaP register (CSTRP)
0016  1662 ;            and compare it to the expected address for the current iteration.
0016  1663 ;        11. Execute micro-code to read back the IR and OSR:
0016  1664 ;                 8. Load the MDR with the IR zero extended.
0016  1665 ;                 9. Load an 8 into the rotator's P-LATCH.
0016  1666 ;                10. Shift the MDR left 8 bits and load it into RTEMP0.
0016  1667 ;                11. Load the MDR with the OSR zero extended.
0016  1668 ;                12. OR the MDR with RTEMP0 and store the results into RTEMP0.
0016  1669 ;        12. If CSTRP does not equal the expected address, signal an error and
0016  1670 ;            dump out RTEMP0.
0016  1671 ;        13. Continue the loop for all modes.
0016  1672 ;        14. Execute the micro-code to place the CPU back into native mode.
0016  1673 ;        15. Close the window to the signature analyzer.
0016  1674 ;        16. End of test
0016  1675 ;
0016  1676 ; TEST DATA
0016  1677 ;
0016  1678 ;       Only the first operand of the two operand MOV instruction is used.
0016  1679 ;
0016  1680 ;       Below is a table that shows the different opcodes, expected control
0016  1681 ;       store addresses, operands and the associated modes used in each loop.
0016  1682 ;
0016  1683 ;       Loop    Opcode  CS Address      Operands        Mode
0016  1684 ;
0016  1685 ;         1.     1000      02D0        R0,R0           REGISTER
0016  1686 ;         2.     11C0      02D1        R7,R0           REGISTER MODE PC
0016  1687 ;         3.     1200      02D2        (R0),R0         REGISTER DEFERRED
```

```
0016  ;          4.     13C0      02D3          (R7),R0          REGISTER DEFERRED PC
0016  ;          5.     1400      02D4          +(R0),R0         AUTO-INCR
0016  ;          6.     1580      02D5          (R6)+,R0         AUTO-INCR STACK POINTER
0016  ;          7.     1600      02D6          @(R0)+,R0        AUTO-INCR DEFERRED
0016  ;          8.     17C0      02D7          @#Addr,R0        AUTO-INCR DEFERRED PC
0016  ;          9.     1800      02D8          -(R0),R0         AUTO-DECR
0016  ;          A.     1980      02D9          -(R6),R0         AUTO-DECR STACK POINTER
0016  ;          B.     1A00      02DA          @-(R0),R0        AUTO-DECR DEFERRED
0016  ;          C.     19C0      02DB          -(R7),R0         AUTO-DECR PC
0016  ;          D.     1C00      02DC          X(R0),R0         INDEX
0016  ;          E.     1DC0      02DD          Addr X(R7),R0    RELATIVE
0016  ;          F.     1E00      02DE          @Addr @X(R0),R0  INDEX DEFERRED
0016  ;         10.     1FC0      02DE          @Addr @X(R7),R0  RELATIVE DEFERRED
0016  ;         11.     15C0      02DF          #,R0             IMMEDIATE
0016  ;
0016  ; LOGICAL DESCRIPTION
0016  ;
0016  ;       The addresses generated by this test come from the Compatibility Mode
0016  ;       IRD1 and IRDX ROMS on the DPM. They are addressed by the IRD gate array
0016  ;       which gets some of its control signals from the PHB gate array.
0016  ;
0016  ; ASSUMPTIONS
0016  ;
0016  ;       This test assumes the following logic is working; the VA, PC, RTEMPS,
0016  ;       ALU, ROTATOR, and the IR and OSR.
0016  ;
0016  ; ERROR DESCRIPTION
0016  ;
0016  ;       On error this test will call out the IRD and PHB gate array and the
0016  ;       following information:
0016  ;
0016  ;       EXPECTED CONTROL STORE ADDRESS
0016  ;       RECEVIED CONTROL STORE ADDRESS
0016  ;       LOOP COUNT
0016  ;       CONTENTS OF THE IR AND OSR
0016  ;--
0016  ;*******************************************************************************
0016  ;///////////////////////////////////////////////////////////////////////////////
0016
0016          INITIALIZE                                  ; INITIALIZE THE CPU
0018          FETCH          0                            ; PUT THE CPU INTO COMP. MODE
001D          BRSTCLK        3                            ; ...
0021          EXPUTRAP                                    ; EXPECT NON-DCS ADDRESSES
0023          LOADREG        WDREG,1$,,L                  ; INITIALIZE THE WD REGISTER
002B          LOOP           I,1,<^X11>                   ; LOOP FOR ALL MODES
0032          LOADREG        WDREG,2$,I,W                 ; LOAD AN OPCODE INTO WDREG
003A          ERRLOOP                                     ; LOOP ON ERROR FROM HERE
003C          FETCH          4                            ; EXECUTE INSTRUCTION DECODE
0041          BRSTCLK        <^X0E>,INH                   ; ...
0045          CMPREGMSK      CSTRP,3$,I,4$,,W             ; IS THE ADDRESS CORRECT?
0051          FETCH          <^X0F>                       ; PUT THE IR/OSR INTO RTEMP0
0056          BRSTCLK        <^X15>,INH                   ; ...
005A          IFERROR        1,<<IRD><PHB>>               ; IF BAD ADDRESS, SIGNAL ERROR
0061          ENDLOOP        I                            ; CONTINUE LOOP
0064          FETCH          <^X16>                       ; PUT CPU BACK INTO NATIVE MODE
0069          BRSTCLK        <^X18>                       ; ...
006D          SKIP                                        ; END OF TEST
```

```
                                0074  1745
                                0074  1746 BEGIN_TEST_DATA
                                0074
                                0074       ;-------------------------------------------------------------------------
                                0074  1747
                     00000000   0074  1748 1$:     .LONG   0       ; DATA TO INITIALIZE THE WD REGISTER
                                0078  1749 ;
                                0078  1750 ; Test Opcodes
                                0078  1751 ;
                                0078  1752 ; Below is the table of test opcodes. The numbers to the right give the loop
                                0078  1753 ; count for each opcode.
                                0078  1754 ;
 1600 1580 1400 13C0 1200 11C0 1000  0078  1755 2$:     .WORD   ^X1000,^X11C0,^X1200,^X13C0,^X1400,^X1580,^X1600 ; 01 - 07
 1DC0 1C00 19C0 1A00 1980 1800 17C0  0086  1756         .WORD   ^X17C0,^X1800,^X1980,^X1A00,^X19C0,^X1C00,^X1DC0 ; 08 - 0E
                   15C0 1FC0 1E00    0094  1757         .WORD   ^X1E00,^X1FC0,^X15C0                             ; 0F - 11
                                009A  1758 ;
                                009A  1759 ; Expected Addresses
                                009A  1760 ;
                                009A  1761 ; Below is the table of expected control store addresses that should be
                                009A  1762 ; generated by the opcodes above. The numbers to the right indicate the loop
                                009A  1763 ; count for each address.
                                009A  1764 ;
 02D6 02D5 02D4 02D3 02D2 02D1 02D0  009A  1765 3$:     .WORD   ^X02D0,^X02D1,^X02D2,^X02D3,^X02D4,^X02D5,^X02D6 ; 01 - 07
 02DD 02DC 02DB 02DA 02D9 02D8 02D7  00A8  1766         .WORD   ^X02D7,^X02D8,^X02D9,^X02DA,^X02DB,^X02DC,^X02DD ; 08 - 0E
                   02DF 02DE 02DE    00B6  1767         .WORD   ^X02DE,^X02DE,^X02DF                             ; 0F - 11
                                00BC  1768
                         3FFF   00BC  1769 4$:     .WORD   ^X3FFF  ; MASK FOR ADDRESS COMPARISON
                                00BE  1770
                                00BE  1771 BEGIN_CPU_DATA
                                0002  1772
                                0002  1773 DCS_DATA
                                0001  1774
U 00, 7700,C800,0364,0300,0470,1801  0001  1775 ;% 00   NOP                                     ; GET THE CLOCK UP TO SPEED
U 01, 7700,3800,3FFF,FFFF,8470,1802  000C  1779 ;% 01   LONLIT_[80000000]                       ; DATA TO SET CM BIT IN PSL
U 02, 7700,C800,5BE4,03D4,0070,1803  0017  1783 ;% 02   WB_R[LONLIT],CCPSL/PSL_WB.CCBR_ALUS    ; PUT CPU INTO COMP. MODE
U 03, 7300,C800,0364,0300,0470,1804  0022  1787 ;% 03   STOP.CLOCK                             ; HALT
                                002D  1791
U 04, 7700,4800,0364,0300,0470,1805  002D  1792 ;% 04   NOP                                     ; GET THE CLOCK UP TO SPEED
U 05, 5700,0840,0364,0300,0470,1806  0038  1796 ;% 05   R[TEMP0]_RDM                            ; PUT OPCODE INTO RTEMP0
U 06, 7700,C800,5BE4,03D8,4A70,1807  0043  1800 ;% 06   VA_R[ZERO]                              ; ZERO THE VA
U 07, 7701,8800,5BE4,0200,5590,1808  004E  1804 ;% 07   WRITE.LONG R[TEMP0]                     ; WRITE OPCODE TO CACHE
U 08, 7701,0800,5BE4,0200,55F0,1809  0059  1808 ;% 08   WDR_UNROT(R[TEMP0]),BUS/PRB.RD,SIZE[LONG]
                                0064  1812                                                    ; ...
U 09, 7700,C800,0364,0300,4470,180A  0064  1813 ;% 09   VA_VA+4                                ; INCREMENT THE VA
U 0A, 7701,8800,5BE4,02D8,5590,180B  006F  1817 ;% 0A   WRITE.LONG R[ZERO]                     ; VALIDATE CACHE FOR XB1
U 0B, 7701,0800,5BE4,02D8,55F0,180C  007A  1821 ;% 0B   WDR_UNROT(R[ZERO]),BUS/PRB.RD,SIZE[LONG]
                                0085  1825                                                    ; ...
U 0C, 7700,4800,5BE4,03D8,4870,180D  0085  1826 ;% 0C   PC_R[ZERO]                             ; INITIATE INST. PREFETCH
U 0D, 7700,8800,0364,1300,0470,180E  0090  1830 ;% 0D   BUT/IRD1                               ; DECODE THE INSTRUCTION
U 0E, 7300,4800,0364,0300,0470,180F  009B  1834 ;% 0E   STOP.CLOCK                             ; HALT
                                00A6  1838
U 0F, 7700,C800,0364,0300,0470,1810  00A6  1839 ;% 0F   NOP                                     ; GET THE CLOCK UP TO SPEED
U 10, 7700,C800,0364,0300,5670,1811  00B1  1843 ;% 10   WCTRL/MDR_IR                            ; LOAD THE MDR WITH THE IR
U 11, 7700,D800,EF64,0304,0470,1812  00BC  1847 ;% 11   PL_[8]                                 ; PUT THE OPCODE INTO THE
U 12, 7700,8852,8770,0300,0470,1813  00C7  1851 ;% 12   R[TEMP0]_M[MDR].RL.P                   ; 2ND BYTE OF THE LONGWORD
U 13, 7700,4800,0364,0300,5E70,1814  00D2  1855 ;% 13   CCPSL/MDR_OSR.CCBR_BRATST              ; LOAD THE MDR WITH THE OSR
U 14, 7700,C852,0024,0300,0470,1815  00DD  1859 ;% 14   R[TEMP0]_M[MDR].OR.RB                  ; RTEMP0 = MODEL OPCODE
```

```
U 15, 7300,C800,0364,0300,0470,1816  00E8  1863 ;% 15   STOP.CLOCK                                  ; HALT
                                     00F3  1867
U 16, 7700,4800,0364,0300,0470,1817  00F3  1868 ;% 16   NOP                                         ; GET THE CLOCK UP TO SPEED
U 17, 7700,C800,5BE4,03D8,0070,1818  00FE  1872 ;% 17   WB_R[ZERO],CCPSL/PSL_WB.CCBR_ALUS          ; RESTORE CPU TO NATIVE MODE
U 18, 7300,C800,0364,0300,0470,1819  0109  1876 ;% 18   STOP.CLOCK                                  ; HALT
                                     0114  1880
                                     0114  1881 RTEMP_DATA
                                     0001  1882
                           00000000  0001  1883          .LONG   0               ; MODEL OPCODE
                                     0005  1884
                                     0005  1885 END_CPU_DATA
                                     00BE  1886
                                     00BE  1887 END_TEST_DATA
                                     00BE
                                     00BE        ;-------------------------------------------------------------------------
                                     00BE  1888
                                     00BE  1889 ENDTEST
```

```
0028        .SBTTL  "TEST   0D4     COMPATABILITY MODE DSIZE ROM TEST 1
0028  1891 ;*******************************************************************************
0028  1892 ;++
0028  1893 ;
0028  1894 ; FUNCTIONAL DESCRIPTION:
0028  1895 ;
0028  1896 ;       THIS IS ONE IN A SERIES OF 13 TESTS THAT CHECK THE CONTENTS OF THE
0028  1897 ;       DSIZE ROM.
0028  1898 ;
0028  1899 ;
0028  1900 ; TEST ALGORITHM:
0028  1901 ;
0028  1902 ;       1.  CREATE A TEST LOOP OF 50
0028  1903 ;       2.  SAVE LOOP COUNT IN 1$
0028  1904 ;       3.  LOAD LOOP COUNT INTO WDREG
0028  1905 ;       4.  EXECUTE DCS PROGRAM
0028  1906 ;               a.  DISABLE ^P INTERRUPTS
0028  1907 ;               b.  PUT CPU IN COMPATABILITY MODE WITH IPL = 1F
0028  1908 ;               c.  MOVE LOOP COUNT TO RTEMP3
0028  1909 ;               d.  ROTATE LOOP COUNT LEFT BY 2 TO FORM A CACHE ADDRESS
0028  1910 ;               e.  LOAD THIS ADDRESS INTO THE PC TO CAUSE PREFETCH
0028  1911 ;               f.  PERFORM IRD1TEST TO LOAD INSTRUCTION REGISTER
0028  1912 ;               h.  SET DTYPE TO INSTRUCTION DEPENDENT AND STROBE THE VBUS
0028  1913 ;       5.  TEST VBUS FOR CORRECT DSIZE
0028  1914 ;       6.  IF NO ERROR GO TO STEP 2 UNTIL LOOP I IS EXHAUSTED
0028  1915 ;
0028  1916 ;
0028  1917 ; TEST PATTERNS:
0028  1918 ;
0028  1919 ;       LOOP COUNT I            INSTRUCTION
0028  1920 ;            1                    ADC
0028  1921 ;            2                    ADCB
0028  1922 ;            3                    ADD,1
0028  1923 ;            4                    ADD,2
0028  1924 ;            5                    ASH,1
0028  1925 ;            6                    ASH,2
0028  1926 ;            7                    ASHC,1
0028  1927 ;            8                    ASHC,2
0028  1928 ;            9                    ASL
0028  1929 ;            A                    ASLB
0028  1930 ;            B                    ASR
0028  1931 ;            C                    ASRB
0028  1932 ;            D                    BCC/BHIS
0028  1933 ;            E                    BCS/BLO
0028  1934 ;            F                    BEQ
0028  1935 ;           10                    BGE
0028  1936 ;           11                    BGT
0028  1937 ;           12                    BHI
0028  1938 ;           13                    BIC,1
0028  1939 ;           14                    BIC,2
0028  1940 ;           15                    BICB,1
0028  1941 ;           16                    BICB,2
0028  1942 ;           17                    BIS,1
0028  1943 ;           18                    BIS,2
0028  1944 ;           19                    BISB,1
0028  1945 ;           1A                    BISB,2
0028  1946 ;           1B                    BIT,1
```

```
                0028  1947 ;          1C                      BIT,2
                0028  1948 ;          1D                      BITB,1
                0028  1949 ;          1E                      BITB,2
                0028  1950 ;          1F                      BLE
                0028  1951 ;          20                      BLOS
                0028  1952 ;          21                      BLT
                0028  1953 ;          22                      BMI
                0028  1954 ;          23                      BNE
                0028  1955 ;          24                      BPL
                0028  1956 ;          25                      BPT
                0028  1957 ;          26                      BR
                0028  1958 ;          27                      BVC
                0028  1959 ;          28                      BVS
                0028  1960 ;          29                      CC-CLR
                0028  1961 ;          2A                      CC-SET
                0028  1962 ;          2B                      CLR
                0028  1963 ;          2C                      CLRB
                0028  1964 ;          2D                      CMP,1
                0028  1965 ;          2E                      CMP,2
                0028  1966 ;          2F                      CMPB,1
                0028  1967 ;          30                      CMPB,2
                0028  1968 ;          31                      COM
                0028  1969 ;          32                      COMB
                0028  1970 ;
                0028  1971 ;
                0028  1972 ; LOGIC DESCRIPTION:
                0028  1973 ;
                0028  1974 ;       THIS TEST ALONG WITH THE OTHER 12 TESTS IN THIS SERIES CHECK ALL THE
                0028  1975 ;       LOCATIONS BLASTED INTO THE DSIZE ROM.  ANY FALIURES SHOULD BE CONFINED
                0028  1976 ;       TO THE ROM.
                0028  1977 ;
                0028  1978 ;
                0028  1979 ; ASSUMPTIONS:
                0028  1980 ;
                0028  1981 ;       THIS TEST ASSUMES THE PREFETCH LOGIC AND INSTRUCTION REGISTER ARE
                0028  1982 ;       OPERATIONAL.
                0028  1983 ;
                0028  1984 ;
                0028  1985 ; ERROR DESCRIPTION:
                0028  1986 ;
                0028  1987 ;       EXPECTED VBUS DATA (DSIZE)
                0028  1988 ;       RECEIVED VBUS DATA
                0028  1989 ;       LOOP COUNT I
                0028  1990 ;               (BITS <7:0> = VBUS BIT POSITION)
                0028  1991 ;               (BITS <8> = BIT STATE)
                0028  1992 ;
                0028  1993 ;--
                0028  1994 ;*****************************************************************************
                0028  1995
                0028  1996
                0028  1997          INITIALIZE                          ;INIT THE CPU
                002A  1998          LOOP            I,1,50              ;CREATE A TEST LOOP OF 50
                0031  1999          SAVEINDEX       I,1$                ;SAVE LOOP COUNT IN 1$
                0036  2000          LOADREG         WDREG,1$,,L         ;LOAD LOOP COUNT INTO WDREG
                003E  2001          ERRLOOP                             ;ERROR LOOP POINT
                0040  2002          FETCH           0                   ;START DCS PROGRAM
                0045  2003          BRSTCLK         <^X0A>              ;END DCS PROGRAM
```

```
                 0049  2004          CMPVBUS         2$,I                    ;TEST DSIZE
                 004E  2005          IFERROR                                 ;ROM FAILURE
                 0053  2006          ENDLOOP         I                       ;END TEST LOOP I
                 0056  2007          SKIP                                    ;GO TO NEXT TEST
                 005D  2008
                 005D  2009
                 005D  2010 BEGIN_TEST_DATA
                 005D
                 005D       ;-----------------------------------------------------------------------
                 005D  2011
        00000000 005D  2012 1$:      .LONG   ^X00000000                      ;LOOP COUNT STORAGE
              02 0061  2013 2$:      .BYTE   ^X2                             ;RESULTS TABLE
                 0062  2014          VBUSDATA        <^X24>,1                ;ADC
                 0063  2015          VBUSDATA        <^X25>,0
              02 0064  2016          .BYTE   ^X2                             ;ADCB
                 0065  2017          VBUSDATA        <^X24>,0
                 0066  2018          VBUSDATA        <^X25>,0
              02 0067  2019          .BYTE   ^X2                             ;ADD,1
                 0068  2020          VBUSDATA        <^X24>,1
                 0069  2021          VBUSDATA        <^X25>,0
              02 006A  2022          .BYTE   ^X2                             ;ADD,2
                 006B  2023          VBUSDATA        <^X24>,1
                 006C  2024          VBUSDATA        <^X25>,0
              02 006D  2025          .BYTE   ^X2                             ;ASH,1
                 006E  2026          VBUSDATA        <^X24>,1
                 006F  2027          VBUSDATA        <^X25>,0
              02 0070  2028          .BYTE   ^X2                             ;ASH,2
                 0071  2029          VBUSDATA        <^X24>,1
                 0072  2030          VBUSDATA        <^X25>,0
              02 0073  2031          .BYTE   ^X2                             ;ASHC,1
                 0074  2032          VBUSDATA        <^X24>,1
                 0075  2033          VBUSDATA        <^X25>,0
              02 0076  2034          .BYTE   ^X2                             ;ASHC,2
                 0077  2035          VBUSDATA        <^X24>,1
                 0078  2036          VBUSDATA        <^X25>,0
              02 0079  2037          .BYTE   ^X2                             ;ASL
                 007A  2038          VBUSDATA        <^X24>,1
                 007B  2039          VBUSDATA        <^X25>,0
              02 007C  2040          .BYTE   ^X2                             ;ASLB
                 007D  2041          VBUSDATA        <^X24>,0
                 007E  2042          VBUSDATA        <^X25>,0
              02 007F  2043          .BYTE   ^X2                             ;ASR
                 0080  2044          VBUSDATA        <^X24>,1
                 0081  2045          VBUSDATA        <^X25>,0
              02 0082  2046          .BYTE   ^X2                             ;ASRB
                 0083  2047          VBUSDATA        <^X24>,0
                 0084  2048          VBUSDATA        <^X25>,0
              02 0085  2049          .BYTE   ^X2                             ;BCC/BHIS
                 0086  2050          VBUSDATA        <^X24>,0
                 0087  2051          VBUSDATA        <^X25>,0
              02 0088  2052          .BYTE   ^X2                             ;BCS/BLO
                 0089  2053          VBUSDATA        <^X24>,0
                 008A  2054          VBUSDATA        <^X25>,0
              02 008B  2055          .BYTE   ^X2                             ;BEQ
                 008C  2056          VBUSDATA        <^X24>,0
                 008D  2057          VBUSDATA        <^X25>,0
              02 008E  2058          .BYTE   ^X2                             ;BGE
```

```
                    008F  2059          VBUSDATA        <^X24>,0
                    0090  2060          VBUSDATA        <^X25>,0
                02  0091  2061          .BYTE   ^X2                             ;BGT
                    0092  2062          VBUSDATA        <^X24>,0
                    0093  2063          VBUSDATA        <^X25>,0
                02  0094  2064          .BYTE   ^X2                             ;BHI
                    0095  2065          VBUSDATA        <^X24>,0
                    0096  2066          VBUSDATA        <^X25>,0
                02  0097  2067          .BYTE   ^X2                             ;BIC,1
                    0098  2068          VBUSDATA        <^X24>,1
                    0099  2069          VBUSDATA        <^X25>,0
                02  009A  2070          .BYTE   ^X2                             ;BIC,2
                    009B  2071          VBUSDATA        <^X24>,1
                    009C  2072          VBUSDATA        <^X25>,0
                02  009D  2073          .BYTE   ^X2                             ;BICB,1
                    009E  2074          VBUSDATA        <^X24>,0
                    009F  2075          VBUSDATA        <^X25>,0
                02  00A0  2076          .BYTE   ^X2                             ;BICB,2
                    00A1  2077          VBUSDATA        <^X24>,0
                    00A2  2078          VBUSDATA        <^X25>,0
                02  00A3  2079          .BYTE   ^X2                             ;BIS,1
                    00A4  2080          VBUSDATA        <^X24>,1
                    00A5  2081          VBUSDATA        <^X25>,0
                02  00A6  2082          .BYTE   ^X2                             ;BIS,2
                    00A7  2083          VBUSDATA        <^X24>,1
                    00A8  2084          VBUSDATA        <^X25>,0
                02  00A9  2085          .BYTE   ^X2                             ;BISB,1
                    00AA  2086          VBUSDATA        <^X24>,0
                    00AB  2087          VBUSDATA        <^X25>,0
                02  00AC  2088          .BYTE   ^X2                             ;BISB,2
                    00AD  2089          VBUSDATA        <^X24>,0
                    00AE  2090          VBUSDATA        <^X25>,0
                02  00AF  2091          .BYTE   ^X2                             ;BIT,1
                    00B0  2092          VBUSDATA        <^X24>,1
                    00B1  2093          VBUSDATA        <^X25>,0
                02  00B2  2094          .BYTE   ^X2                             ;BIT,2
                    00B3  2095          VBUSDATA        <^X24>,1
                    00B4  2096          VBUSDATA        <^X25>,0
                02  00B5  2097          .BYTE   ^X2                             ;BITB,1
                    00B6  2098          VBUSDATA        <^X24>,0
                    00B7  2099          VBUSDATA        <^X25>,0
                02  00B8  2100          .BYTE   ^X2                             ;BITB,2
                    00B9  2101          VBUSDATA        <^X24>,0
                    00BA  2102          VBUSDATA        <^X25>,0
                02  00BB  2103          .BYTE   ^X2                             ;BLE
                    00BC  2104          VBUSDATA        <^X24>,0
                    00BD  2105          VBUSDATA        <^X25>,0
                02  00BE  2106          .BYTE   ^X2                             ;BLOS
                    00BF  2107          VBUSDATA        <^X24>,0
                    00C0  2108          VBUSDATA        <^X25>,0
                02  00C1  2109          .BYTE   ^X2                             ;BLT
                    00C2  2110          VBUSDATA        <^X24>,0
                    00C3  2111          VBUSDATA        <^X25>,0
                02  00C4  2112          .BYTE   ^X2                             ;BMI
                    00C5  2113          VBUSDATA        <^X24>,0
                    00C6  2114          VBUSDATA        <^X25>,0
                02  00C7  2115          .BYTE   ^X2                             ;BNE
```

```
                    00C8  2116          VBUSDATA        <^X24>,0
                    00C9  2117          VBUSDATA        <^X25>,0
                 02 00CA  2118          .BYTE   ^X2                             ;BPL
                    00CB  2119          VBUSDATA        <^X24>,0
                    00CC  2120          VBUSDATA        <^X25>,0
                 02 00CD  2121          .BYTE   ^X2                             ;BPT
                    00CE  2122          VBUSDATA        <^X24>,0
                    00CF  2123          VBUSDATA        <^X25>,0
                 02 00D0  2124          .BYTE   ^X2                             ;BR
                    00D1  2125          VBUSDATA        <^X24>,0
                    00D2  2126          VBUSDATA        <^X25>,0
                 02 00D3  2127          .BYTE   ^X2                             ;BVC
                    00D4  2128          VBUSDATA        <^X24>,0
                    00D5  2129          VBUSDATA        <^X25>,0
                 02 00D6  2130          .BYTE   ^X2                             ;BVS
                    00D7  2131          VBUSDATA        <^X24>,0
                    00D8  2132          VBUSDATA        <^X25>,0
                 02 00D9  2133          .BYTE   ^X2                             ;CC-CLR
                    00DA  2134          VBUSDATA        <^X24>,1
                    00DB  2135          VBUSDATA        <^X25>,0
                 02 00DC  2136          .BYTE   ^X2                             ;CC-SET
                    00DD  2137          VBUSDATA        <^X24>,1
                    00DE  2138          VBUSDATA        <^X25>,0
                 02 00DF  2139          .BYTE   ^X2                             ;CLR
                    00E0  2140          VBUSDATA        <^X24>,1
                    00E1  2141          VBUSDATA        <^X25>,0
                 02 00E2  2142          .BYTE   ^X2                             ;CLRB
                    00E3  2143          VBUSDATA        <^X24>,0
                    00E4  2144          VBUSDATA        <^X25>,0
                 02 00E5  2145          .BYTE   ^X2                             ;CMP,1
                    00E6  2146          VBUSDATA        <^X24>,1
                    00E7  2147          VBUSDATA        <^X25>,0
                 02 00E8  2148          .BYTE   ^X2                             ;CMP,2
                    00E9  2149          VBUSDATA        <^X24>,1
                    00EA  2150          VBUSDATA        <^X25>,0
                 02 00EB  2151          .BYTE   ^X2                             ;CMPB,1
                    00EC  2152          VBUSDATA        <^X24>,0
                    00ED  2153          VBUSDATA        <^X25>,0
                 02 00EE  2154          .BYTE   ^X2                             ;CMPB,2
                    00EF  2155          VBUSDATA        <^X24>,0
                    00F0  2156          VBUSDATA        <^X25>,0
                 02 00F1  2157          .BYTE   ^X2                             ;COM
                    00F2  2158          VBUSDATA        <^X24>,1
                    00F3  2159          VBUSDATA        <^X25>,0
                 02 00F4  2160          .BYTE   ^X2                             ;COMB
                    00F5  2161          VBUSDATA        <^X24>,0
                    00F6  2162          VBUSDATA        <^X25>,0
                    00F7  2163
                    00F7  2164
                    00F7  2165 BEGIN_CPU_DATA
                    0002  2166
                    0002  2167 DCS_DATA
                    0001  2168
U 00, 7700,C800,0364,0300,0470,1801 0001  2169 ;% 00   NOP
U 01, 7700,4800,5BE4,0300,2470,1802 000C  2173 ;% 01   WCTRL/LOADCRAR,WB_R[TEMP0]          ;DISABLE ^P INTERRUPT
U 02, 7700,0800,5BE4,0304,2C70,1803 0017  2177 ;% 02   WCTRL/CONWRITE,WB_R[TEMP1]         ;...
U 03, 7700,4800,5BF4,0308,0070,1804 0022  2181 ;% 03   PSL_R[TEMP2]                       ;SET CM AND IPL = 1F
```

```
U 04, 5700,0840,0364,030C,0470,1805  002D  2185 ;% 04   R[TEMP3]_RDM                          ;MOVE LOOP COUNT TO RTEMP2
U 05, 7700,D800,EF64,0301,0470,1806  0038  2189 ;% 05   PL_[2]                               ;SET PLATCH TO 2
U 06, 7700,8800,8B71,030C,0470,1807  0043  2193 ;% 06   Q_R[TEMP3].RL.P                      ;ROTATE LOOP COUNT LEFT 2
U 07, 7700,4800,03A4,03D8,4870,1808  004E  2197 ;% 07   WCTRL/PC_WB,WB_Q                     ;FORCE PREFETCH TO HAPPEN
U 08, 7700,4800,0364,1700,0470,1809  0059  2201 ;% 08   IRD1TEST                            ;LOAD IR REGISTER
U 09, 7600,4800,0364,0300,0470,180A  0064  2205 ;% 09   DTYPE/IDEP,STROBE.V.BUS             ;LOAD DSIZE INTO VBUS
U 0A, 7300,C800,0364,0300,0470,180B  006F  2209 ;% 0A   NOP,STOP.CLOCK                      ;STOP THE CPU CLOCK
                                     007A  2213
                                     007A  2214 RTEMP_DATA
                                     0001  2215
                           00C00000  0001  2216         .LONG   ^X00C00000                ;DISABLE ^P INTERRUPT
                           00400000  0005  2217         .LONG   ^X00400000                ;...
                           801F0000  0009  2218         .LONG   ^X801F0000                ;PSL
                                     000D  2219
                                     000D  2220 CACHE_DATA
                                     0001  2221
                           00000B40  0001  2222         .LONG   ^X00000B40                ;ADC
                           00008B40  0005  2223         .LONG   ^X00008B40                ;ADCB
                           00006000  0009  2224         .LONG   ^X00006000                ;ADD,1
                           00006080  000D  2225         .LONG   ^X00006080                ;ADD,2
                           00007400  0011  2226         .LONG   ^X00007400                ;ASH,1
                           00007408  0015  2227         .LONG   ^X00007408                ;ASH,2
                           00007600  0019  2228         .LONG   ^X00007600                ;ASHC,1
                           00007608  001D  2229         .LONG   ^X00007608                ;ASHC,2
                           00000CC0  0021  2230         .LONG   ^X00000CC0                ;ASL
                           00008CC0  0025  2231         .LONG   ^X00008CC0                ;ASLB
                           00000C80  0029  2232         .LONG   ^X00000C80                ;ASR
                           00008C80  002D  2233         .LONG   ^X00008C80                ;ASRB
                           00008600  0031  2234         .LONG   ^X00008600                ;BCC/BHIS
                           00008700  0035  2235         .LONG   ^X00008700                ;BCS/BLO
                           00000300  0039  2236         .LONG   ^X00000300                ;BEQ
                           00000400  003D  2237         .LONG   ^X00000400                ;BGE
                           00000600  0041  2238         .LONG   ^X00000600                ;BGT
                           00008200  0045  2239         .LONG   ^X00008200                ;BHI
                           00004000  0049  2240         .LONG   ^X00004000                ;BIC,1
                           00004080  004D  2241         .LONG   ^X00004080                ;BIC,2
                           0000C000  0051  2242         .LONG   ^X0000C000                ;BICB,1
                           0000C080  0055  2243         .LONG   ^X0000C080                ;BICB,2
                           00005000  0059  2244         .LONG   ^X00005000                ;BIS,1
                           00005080  005D  2245         .LONG   ^X00005080                ;BIS,2
                           0000D000  0061  2246         .LONG   ^X0000D000                ;BISB,1
                           0000D080  0065  2247         .LONG   ^X0000D080                ;BISB,2
                           00003000  0069  2248         .LONG   ^X00003000                ;BIT,1
                           00003080  006D  2249         .LONG   ^X00003080                ;BIT,2
                           0000B000  0071  2250         .LONG   ^X0000B000                ;BITB,1
                           0000B080  0075  2251         .LONG   ^X0000B080                ;BITB,1
                           00000700  0079  2252         .LONG   ^X00000700                ;BLE
                           00008300  007D  2253         .LONG   ^X00008300                ;BLOS
                           00000500  0081  2254         .LONG   ^X00000500                ;BLT
                           00008100  0085  2255         .LONG   ^X00008100                ;BMI
                           00000200  0089  2256         .LONG   ^X00000200                ;BNE
                           00008600  008D  2257         .LONG   ^X00008600                ;BPL
                           00000003  0091  2258         .LONG   ^X00000003                ;BPT
                           00000100  0095  2259         .LONG   ^X00000100                ;BR
                           00008400  0099  2260         .LONG   ^X00008400                ;BVC
                           00008500  009D  2261         .LONG   ^X00008500                ;BVS
                           000000A0  00A1  2262         .LONG   ^X000000A0                ;CC-CLR
```

```
000000B0  00A5  2263          .LONG   ^X000000B0                      ;CC-SET
00000A00  00A9  2264          .LONG   ^X00000A00                      ;CLR
00008A00  00AD  2265          .LONG   ^X00008A00                      ;CLRB
00002000  00B1  2266          .LONG   ^X00002000                      ;CMP,1
00002080  00B5  2267          .LONG   ^X00002080                      ;CMP,2
0000A000  00B9  2268          .LONG   ^X0000A000                      ;CMPB,1
0000A080  00BD  2269          .LONG   ^X0000A080                      ;CMPB,2
00000A40  00C1  2270          .LONG   ^X00000A40                      ;COM
00008A40  00C5  2271          .LONG   ^X00008A40                      ;COMB
          00C9  2272
          00C9  2273 END_CPU_DATA
          00F7  2274
          00F7  2275
          00F7  2276 END_TEST_DATA
          00F7
          00F7          ;-----------------------------------------------------------------------------
          00F7  2277 ENDTEST
```

```
0028          .SBTTL  "TEST   0D5     COMPATABILITY MODE DSIZE ROM TEST 2
0028  2279 ;*******************************************************************************
0028  2280 ;++
0028  2281 ;
0028  2282 ; FUNCTIONAL DESCRIPTION:
0028  2283 ;
0028  2284 ;       THIS IS ONE IN A SERIES OF 13 TESTS THAT CHECK THE CONTENTS OF THE
0028  2285 ;       DSIZE ROM.
0028  2286 ;
0028  2287 ;
0028  2288 ; TEST ALGORITHM:
0028  2289 ;
0028  2290 ;       1.  CREATE A TEST LOOP OF 61
0028  2291 ;       2.  SAVE LOOP COUNT IN 1$
0028  2292 ;       3.  LOAD LOOP COUNT INTO WDREG
0028  2293 ;       4.  EXECUTE DCS PROGRAM
0028  2294 ;               a.  DISABLE ^P INTERRUPTS
0028  2295 ;               b.  PUT CPU IN COMPATABILITY MODE WITH IPL = 1F
0028  2296 ;               c.  MOVE LOOP COUNT TO RTEMP3
0028  2297 ;               d.  ROTATE LOOP COUNT LEFT BY 2 TO FORM A CACHE ADDRESS
0028  2298 ;               e.  LOAD THIS ADDRESS INTO THE PC TO CAUSE PREFETCH
0028  2299 ;               f.  PERFORM IRD1TEST TO LOAD INSTRUCTION REGISTER
0028  2300 ;               h.  SET DTYPE TO INSTRUCTION DEPENDENT AND STROBE THE VBUS
0028  2301 ;       5.  TEST VBUS FOR CORRECT DSIZE
0028  2302 ;       6.  IF NO ERROR GO TO STEP 2 UNTIL LOOP I IS EXHAUSTED
0028  2303 ;
0028  2304 ;
0028  2305 ; TEST PATTERNS:
0028  2306 ;
0028  2307 ;       LOOP COUNT I            INSTRUCTION
0028  2308 ;            1                    DEC
0028  2309 ;            2                    DECB
0028  2310 ;            3                    DIV,1
0028  2311 ;            4                    DIV,2
0028  2312 ;            5                    EMT,1
0028  2313 ;            6                    EMT,2
0028  2314 ;            7                    EMT,3
0028  2315 ;            8                    EMT,4
0028  2316 ;            9                    INC
0028  2317 ;            A                    INCB
0028  2318 ;            B                    IOT
0028  2319 ;            C                    JMP,1
0028  2320 ;            D                    JMP,2
0028  2321 ;            E                    JMP,3
0028  2322 ;            F                    JMP,4
0028  2323 ;           10                    JSR,1
0028  2324 ;           11                    JSR,2
0028  2325 ;           12                    JSR,3
0028  2326 ;           13                    JSR,4
0028  2327 ;           14                    JSR,5
0028  2328 ;           15                    JSR,6
0028  2329 ;           16                    JSR,7
0028  2330 ;           17                    JSR,8
0028  2331 ;           18                    MFPD
0028  2332 ;           19                    MFPI
0028  2333 ;           1A                    MTPD
0028  2334 ;           1B                    MTPI
```

```
0028  2335 ;          1C                      MOV,1
0028  2336 ;          1D                      MOV,2
0028  2337 ;          1E                      MOVB,1
0028  2338 ;          1F                      MOVB,2
0028  2339 ;          20                      MUL,1
0028  2340 ;          21                      MUL,2
0028  2341 ;          22                      NEG
0028  2342 ;          23                      NEGB
0028  2343 ;          24                      ROL
0028  2344 ;          25                      ROLB
0028  2345 ;          26                      ROR
0028  2346 ;          27                      RORB
0028  2347 ;          28                      RTI
0028  2348 ;          29                      RTS
0028  2349 ;          2A                      RTT
0028  2350 ;          2B                      SBC
0028  2351 ;          2C                      SBCB
0028  2352 ;          2D                      SOB,1
0028  2353 ;          2E                      SOB,2
0028  2354 ;          2F                      SUB,1
0028  2355 ;          30                      SUB,2
0028  2356 ;          31                      SWAB,1
0028  2357 ;          32                      SWAB,2
0028  2358 ;          33                      SWAB,3
0028  2359 ;          34                      SWAB,4
0028  2360 ;          35                      SXT
0028  2361 ;          36                      TRAP,1
0028  2362 ;          37                      TRAP,2
0028  2363 ;          38                      TRAP,3
0028  2364 ;          39                      TRAP,4
0028  2365 ;          3A                      TST
0028  2366 ;          3B                      TSTB
0028  2367 ;          3C                      XOR,1
0028  2368 ;          3D                      XOR,2
0028  2369 ;
0028  2370 ;
0028  2371 ; LOGIC DESCRIPTION:
0028  2372 ;
0028  2373 ;       THIS TEST ALONG WITH THE OTHER 12 TESTS IN THIS SERIES CHECK ALL THE
0028  2374 ;       LOCATIONS BLASTED INTO THE DSIZE ROM.  ANY FALIURES SHOULD BE CONFINED
0028  2375 ;       TO THE ROM.
0028  2376 ;
0028  2377 ;
0028  2378 ; ASSUMPTIONS:
0028  2379 ;
0028  2380 ;       THIS TEST ASSUMES THE PREFETCH LOGIC AND INSTRUCTION REGISTER ARE
0028  2381 ;       OPERATIONAL.
0028  2382 ;
0028  2383 ;
0028  2384 ; ERROR DESCRIPTION:
0028  2385 ;
0028  2386 ;       EXPECTED VBUS DATA (DSIZE)
0028  2387 ;       RECEIVED VBUS DATA
0028  2388 ;       LOOP COUNT I
0028  2389 ;               (BITS <7:0> = VBUS BIT POSITION)
0028  2390 ;               (BITS <8> = BIT STATE)
0028  2391 ;
```

```
                        0028  2392 ;--
                        0028  2393 ;*****************************************************************************
                        0028  2394
                        0028  2395
                        0028  2396         INITIALIZE                                  ;INIT THE CPU
                        002A  2397         LOOP            I,1,61                      ;CREATE A TEST LOOP OF 50
                        0031  2398         SAVEINDEX       I,1$                        ;SAVE LOOP COUNT IN 1$
                        0036  2399         LOADREG         WDREG,1$,,L                 ;LOAD LOOP COUNT INTO WDREG
                        003E  2400         ERRLOOP                                     ;ERROR LOOP POINT
                        0040  2401         FETCH           0                           ;START DCS PROGRAM
                        0045  2402         BRSTCLK         <^X0A>                      ;END DCS PROGRAM
                        0049  2403         CMPVBUS         2$,I                        ;TEST DSIZE
                        004E  2404         IFERROR                                     ;ROM FAILURE
                        0053  2405         ENDLOOP         I                           ;END TEST LOOP I
                        0056  2406         SKIP                                        ;GO TO NEXT TEST
                        005D  2407
                        005D  2408
                        005D  2409 BEGIN_TEST_DATA
                        005D               ;-------------------------------------------------------------------
                        005D  2410
                00000000  005D  2411 1$:      .LONG   ^X00000000                      ;LOOP COUNT STORAGE
                      02  0061  2412 2$:      .BYTE   ^X2                             ;RESULTS TABLE
                          0062  2413          VBUSDATA        <^X24>,1                ;DEC
                          0063  2414          VBUSDATA        <^X25>,0
                      02  0064  2415          .BYTE   ^X2                             ;DECB
                          0065  2416          VBUSDATA        <^X24>,0
                          0066  2417          VBUSDATA        <^X25>,0
                      02  0067  2418          .BYTE   ^X2                             ;DIV,1
                          0068  2419          VBUSDATA        <^X24>,1
                          0069  2420          VBUSDATA        <^X25>,0
                      02  006A  2421          .BYTE   ^X2                             ;DIV,2
                          006B  2422          VBUSDATA        <^X24>,1
                          006C  2423          VBUSDATA        <^X25>,0
                      02  006D  2424          .BYTE   ^X2                             ;EMT,1
                          006E  2425          VBUSDATA        <^X24>,0
                          006F  2426          VBUSDATA        <^X25>,0
                      02  0070  2427          .BYTE   ^X2                             ;EMT,2
                          0071  2428          VBUSDATA        <^X24>,0
                          0072  2429          VBUSDATA        <^X25>,0
                      02  0073  2430          .BYTE   ^X2                             ;EMT,3
                          0074  2431          VBUSDATA        <^X24>,0
                          0075  2432          VBUSDATA        <^X25>,0
                      02  0076  2433          .BYTE   ^X2                             ;EMT,4
                          0077  2434          VBUSDATA        <^X24>,0
                          0078  2435          VBUSDATA        <^X25>,0
                      02  0079  2436          .BYTE   ^X2                             ;INC
                          007A  2437          VBUSDATA        <^X24>,1
                          007B  2438          VBUSDATA        <^X25>,0
                      02  007C  2439          .BYTE   ^X2                             ;INCB
                          007D  2440          VBUSDATA        <^X24>,0
                          007E  2441          VBUSDATA        <^X25>,0
                      02  007F  2442          .BYTE   ^X2                             ;IOT
                          0080  2443          VBUSDATA        <^X24>,0
                          0081  2444          VBUSDATA        <^X25>,0
                      02  0082  2445          .BYTE   ^X2                             ;JMP,1
                          0083  2446          VBUSDATA        <^X24>,1
```

```
                0084  2447          VBUSDATA        <^X25>,0
02              0085  2448          .BYTE   ^X2                             ;JMP,2
                0086  2449          VBUSDATA        <^X24>,1
                0087  2450          VBUSDATA        <^X25>,0
02              0088  2451          .BYTE   ^X2                             ;JMP,3
                0089  2452          VBUSDATA        <^X24>,1
                008A  2453          VBUSDATA        <^X25>,0
02              008B  2454          .BYTE   ^X2                             ;JMP,4
                008C  2455          VBUSDATA        <^X24>,1
                008D  2456          VBUSDATA        <^X25>,0
02              008E  2457          .BYTE   ^X2                             ;JSR,1
                008F  2458          VBUSDATA        <^X24>,1
                0090  2459          VBUSDATA        <^X25>,0
02              0091  2460          .BYTE   ^X2                             ;JSR,2
                0092  2461          VBUSDATA        <^X24>,1
                0093  2462          VBUSDATA        <^X25>,0
02              0094  2463          .BYTE   ^X2                             ;JSR,3
                0095  2464          VBUSDATA        <^X24>,1
                0096  2465          VBUSDATA        <^X25>,0
02              0097  2466          .BYTE   ^X2                             ;JSR,4
                0098  2467          VBUSDATA        <^X24>,1
                0099  2468          VBUSDATA        <^X25>,0
02              009A  2469          .BYTE   ^X2                             ;JSR,5
                009B  2470          VBUSDATA        <^X24>,1
                009C  2471          VBUSDATA        <^X25>,0
02              009D  2472          .BYTE   ^X2                             ;JSR,6
                009E  2473          VBUSDATA        <^X24>,1
                009F  2474          VBUSDATA        <^X25>,0
02              00A0  2475          .BYTE   ^X2                             ;JSR,7
                00A1  2476          VBUSDATA        <^X24>,1
                00A2  2477          VBUSDATA        <^X25>,0
02              00A3  2478          .BYTE   ^X2                             ;JSR,8
                00A4  2479          VBUSDATA        <^X24>,1
                00A5  2480          VBUSDATA        <^X25>,0
02              00A6  2481          .BYTE   ^X2                             ;MFPD
                00A7  2482          VBUSDATA        <^X24>,1
                00A8  2483          VBUSDATA        <^X25>,0
02              00A9  2484          .BYTE   ^X2                             ;MFPI
                00AA  2485          VBUSDATA        <^X24>,1
                00AB  2486          VBUSDATA        <^X25>,0
02              00AC  2487          .BYTE   ^X2                             ;MTPD
                00AD  2488          VBUSDATA        <^X24>,1
                00AE  2489          VBUSDATA        <^X25>,0
02              00AF  2490          .BYTE   ^X2                             ;MTPI
                00B0  2491          VBUSDATA        <^X24>,1
                00B1  2492          VBUSDATA        <^X25>,0
02              00B2  2493          .BYTE   ^X2                             ;MOV,1
                00B3  2494          VBUSDATA        <^X24>,1
                00B4  2495          VBUSDATA        <^X25>,0
02              00B5  2496          .BYTE   ^X2                             ;MOV,2
                00B6  2497          VBUSDATA        <^X24>,1
                00B7  2498          VBUSDATA        <^X25>,0
02              00B8  2499          .BYTE   ^X2                             ;MOVB,1
                00B9  2500          VBUSDATA        <^X24>,0
                00BA  2501          VBUSDATA        <^X25>,0
02              00BB  2502          .BYTE   ^X2                             ;MOVB,2
                00BC  2503          VBUSDATA        <^X24>,0
```

```
                00BD  2504          VBUSDATA        <^X25>,0
    02          00BE  2505          .BYTE   ^X2                             ;MUL,1
                00BF  2506          VBUSDATA        <^X24>,1
                00C0  2507          VBUSDATA        <^X25>,0
    02          00C1  2508          .BYTE   ^X2                             ;MUL,2
                00C2  2509          VBUSDATA        <^X24>,1
                00C3  2510          VBUSDATA        <^X25>,0
    02          00C4  2511          .BYTE   ^X2                             ;NEG
                00C5  2512          VBUSDATA        <^X24>,1
                00C6  2513          VBUSDATA        <^X25>,0
    02          00C7  2514          .BYTE   ^X2                             ;NEGB
                00C8  2515          VBUSDATA        <^X24>,0
                00C9  2516          VBUSDATA        <^X25>,0
    02          00CA  2517          .BYTE   ^X2                             ;ROL
                00CB  2518          VBUSDATA        <^X24>,1
                00CC  2519          VBUSDATA        <^X25>,0
    02          00CD  2520          .BYTE   ^X2                             ;ROLB
                00CE  2521          VBUSDATA        <^X24>,0
                00CF  2522          VBUSDATA        <^X25>,0
    02          00D0  2523          .BYTE   ^X2                             ;ROR
                00D1  2524          VBUSDATA        <^X24>,1
                00D2  2525          VBUSDATA        <^X25>,0
    02          00D3  2526          .BYTE   ^X2                             ;RORB
                00D4  2527          VBUSDATA        <^X24>,0
                00D5  2528          VBUSDATA        <^X25>,0
    02          00D6  2529          .BYTE   ^X2                             ;RTI
                00D7  2530          VBUSDATA        <^X24>,1
                00D8  2531          VBUSDATA        <^X25>,0
    02          00D9  2532          .BYTE   ^X2                             ;RTS
                00DA  2533          VBUSDATA        <^X24>,1
                00DB  2534          VBUSDATA        <^X25>,0
    02          00DC  2535          .BYTE   ^X2                             ;RTT
                00DD  2536          VBUSDATA        <^X24>,1
                00DE  2537          VBUSDATA        <^X25>,0
    02          00DF  2538          .BYTE   ^X2                             ;SBC
                00E0  2539          VBUSDATA        <^X24>,1
                00E1  2540          VBUSDATA        <^X25>,0
    02          00E2  2541          .BYTE   ^X2                             ;SBCB
                00E3  2542          VBUSDATA        <^X24>,0
                00E4  2543          VBUSDATA        <^X25>,0
    02          00E5  2544          .BYTE   ^X2                             ;SOB,1
                00E6  2545          VBUSDATA        <^X24>,1
                00E7  2546          VBUSDATA        <^X25>,0
    02          00E8  2547          .BYTE   ^X2                             ;SOB,2
                00E9  2548          VBUSDATA        <^X24>,1
                00EA  2549          VBUSDATA        <^X25>,0
    02          00EB  2550          .BYTE   ^X2                             ;SUB,1
                00EC  2551          VBUSDATA        <^X24>,1
                00ED  2552          VBUSDATA        <^X25>,0
    02          00EE  2553          .BYTE   ^X2                             ;SUB,2
                00EF  2554          VBUSDATA        <^X24>,1
                00F0  2555          VBUSDATA        <^X25>,0
    02          00F1  2556          .BYTE   ^X2                             ;SWAB,1
                00F2  2557          VBUSDATA        <^X24>,1
                00F3  2558          VBUSDATA        <^X25>,0
    02          00F4  2559          .BYTE   ^X2                             ;SWAB,2
                00F5  2560          VBUSDATA        <^X24>,1
```

```
                    00F6  2561          VBUSDATA        <^X25>,0
                 02 00F7  2562          .BYTE   ^X2                             ;SWAB,3
                    00F8  2563          VBUSDATA        <^X24>,1
                    00F9  2564          VBUSDATA        <^X25>,0
                 02 00FA  2565          .BYTE   ^X2                             ;SWAB,4
                    00FB  2566          VBUSDATA        <^X24>,1
                    00FC  2567          VBUSDATA        <^X25>,0
                 02 00FD  2568          .BYTE   ^X2                             ;SXT
                    00FE  2569          VBUSDATA        <^X24>,1
                    00FF  2570          VBUSDATA        <^X25>,0
                 02 0100  2571          .BYTE   ^X2                             ;TRAP,1
                    0101  2572          VBUSDATA        <^X24>,0
                    0102  2573          VBUSDATA        <^X25>,0
                 02 0103  2574          .BYTE   ^X2                             ;TRAP,2
                    0104  2575          VBUSDATA        <^X24>,0
                    0105  2576          VBUSDATA        <^X25>,0
                 02 0106  2577          .BYTE   ^X2                             ;TRAP,3
                    0107  2578          VBUSDATA        <^X24>,0
                    0108  2579          VBUSDATA        <^X25>,0
                 02 0109  2580          .BYTE   ^X2                             ;TRAP,4
                    010A  2581          VBUSDATA        <^X24>,0
                    010B  2582          VBUSDATA        <^X25>,0
                 02 010C  2583          .BYTE   ^X2                             ;TST
                    010D  2584          VBUSDATA        <^X24>,1
                    010E  2585          VBUSDATA        <^X25>,0
                 02 010F  2586          .BYTE   ^X2                             ;TSTB
                    0110  2587          VBUSDATA        <^X24>,0
                    0111  2588          VBUSDATA        <^X25>,0
                 02 0112  2589          .BYTE   ^X2                             ;XOR,1
                    0113  2590          VBUSDATA        <^X24>,1
                    0114  2591          VBUSDATA        <^X25>,0
                 02 0115  2592          .BYTE   ^X2                             ;XOR,2
                    0116  2593          VBUSDATA        <^X24>,1
                    0117  2594          VBUSDATA        <^X25>,0
                    0118  2595
                    0118  2596
                    0118  2597 BEGIN_CPU_DATA
                    0002  2598
                    0002  2599 DCS_DATA
                    0001  2600
U 00, 7700,C800,0364,0300,0470,1801 0001  2601 ;* 00   NOP
U 01, 7700,4800,5BE4,0300,2470,1802 000C  2605 ;* 01   WCTRL/LOADCRAR,WB_R[TEMP0]      ;DISABLE ^P INTERRUPT
U 02, 7700,0800,5BE4,0304,2C70,1803 0017  2609 ;* 02   WCTRL/CONWRITE,WB_R[TEMP1]     ;...
U 03, 7700,4800,5BF4,0308,0070,1804 0022  2613 ;* 03   PSL_R[TEMP2]                   ;SET CM AND IPL = 1F
U 04, 5700,0840,0364,030C,0470,1805 002D  2617 ;* 04   R[TEMP3]_RDM                  ;MOVE LOOP COUNT TO RTEMP2
U 05, 7700,D800,EF64,0301,0470,1806 0038  2621 ;* 05   PL_[2]                       ;SET PLATCH TO 2
U 06, 7700,8800,8B71,030C,0470,1807 0043  2625 ;* 06   Q_R[TEMP3].RL.P             ;ROTATE LOOP COUNT LEFT 2
U 07, 7700,4800,03A4,03D8,4870,1808 004E  2629 ;* 07   WCTRL/PC_WB,WB_Q             ;FORCE PREFETCH TO HAPPEN
U 08, 7700,4800,0364,1700,0470,1809 0059  2633 ;* 08   IRD1TEST                     ;LOAD IR REGISTER
U 09, 7600,4800,0364,0300,0470,180A 0064  2637 ;* 09   DTYPE/IDEP,STROBE.V.BUS      ;LOAD DSIZE INTO VBUS
U 0A, 7300,C800,0364,0300,0470,180B 006F  2641 ;* 0A   NOP,STOP.CLOCK              ;STOP THE CPU CLOCK
                    007A  2645
                    007A  2646 RTEMP_DATA
                    0001  2647
                00C00000  0001  2648          .LONG   ^X00C00000                  ;DISABLE ^P INTERRUPT
                00400000  0005  2649          .LONG   ^X00400000                  ;...
                801F0000  0009  2650          .LONG   ^X801F0000                  ;PSL
```

```
                        000D  2651
                        000D  2652 CACHE_DATA
                        0001  2653
00000AC0                0001  2654          .LONG   ^X00000AC0                      ;DEC
00008AC0                0005  2655          .LONG   ^X00008AC0                      ;DECB
00007200                0009  2656          .LONG   ^X00007200                      ;DIV,1
00007280                000D  2657          .LONG   ^X00007280                      ;DIV,2
00008800                0011  2658          .LONG   ^X00008800                      ;EMT,1
00008840                0015  2659          .LONG   ^X00008840                      ;EMT,2
00008880                0019  2660          .LONG   ^X00008880                      ;EMT,3
000088C0                001D  2661          .LONG   ^X000088C0                      ;EMT,4
00000A80                0021  2662          .LONG   ^X00000A80                      ;INC
00008A80                0025  2663          .LONG   ^X00008A80                      ;INCB
00000004                0029  2664          .LONG   ^X00000004                      ;IOT
00000040                002D  2665          .LONG   ^X00000040                      ;JMP,1
00000050                0031  2666          .LONG   ^X00000050                      ;JMP,2
00000060                0035  2667          .LONG   ^X00000060                      ;JMP,3
00000070                0039  2668          .LONG   ^X00000070                      ;JMP,4
00000800                003D  2669          .LONG   ^X00000800                      ;JSR,1
00000840                0041  2670          .LONG   ^X00000840                      ;JSR,2
00000880                0045  2671          .LONG   ^X00000880                      ;JSR,3
000008C0                0049  2672          .LONG   ^X000008C0                      ;JSR,4
00000900                004D  2673          .LONG   ^X00000900                      ;JSR,5
00000940                0051  2674          .LONG   ^X00000940                      ;JSR,6
00000980                0055  2675          .LONG   ^X00000980                      ;JSR,7
000009C0                0059  2676          .LONG   ^X000009C0                      ;JSR,8
00008D40                005D  2677          .LONG   ^X00008D40                      ;MFPD
00000D40                0061  2678          .LONG   ^X00000D40                      ;MFPI
00008D80                0065  2679          .LONG   ^X00008D80                      ;MTPD
00000D80                0069  2680          .LONG   ^X00000D80                      ;MTPI
00001000                006D  2681          .LONG   ^X00001000                      ;MOV,1
00001200                0071  2682          .LONG   ^X00001200                      ;MOV,2
00009000                0075  2683          .LONG   ^X00009000                      ;MOVB,1
00009200                0079  2684          .LONG   ^X00009200                      ;MOVB,2
00007000                007D  2685          .LONG   ^X00007000                      ;MUL,1
00007080                0081  2686          .LONG   ^X00007080                      ;MUL,2
00000B00                0085  2687          .LONG   ^X00000B00                      ;NEG
00008B00                0089  2688          .LONG   ^X00008B00                      ;NEGB
00000C40                008D  2689          .LONG   ^X00000C40                      ;ROL
00008C40                0091  2690          .LONG   ^X00008C40                      ;ROLB
00000C00                0095  2691          .LONG   ^X00000C00                      ;ROR
00008C00                0099  2692          .LONG   ^X00008C00                      ;RORB
00000002                009D  2693          .LONG   ^X00000002                      ;RTI
00000080                00A1  2694          .LONG   ^X00000080                      ;RTS
00000006                00A5  2695          .LONG   ^X00000006                      ;RTT
00000B80                00A9  2696          .LONG   ^X00000B80                      ;SBC
00008B80                00AD  2697          .LONG   ^X00008B80                      ;SBCB
00007E00                00B1  2698          .LONG   ^X00007E00                      ;SOB,1
00007E80                00B5  2699          .LONG   ^X00007E80                      ;SOB,2
0000E000                00B9  2700          .LONG   ^X0000E000                      ;SUB,1
0000E080                00BD  2701          .LONG   ^X0000E080                      ;SUB,2
000000C0                00C1  2702          .LONG   ^X000000C0                      ;SWAB,1
000000D0                00C5  2703          .LONG   ^X000000D0                      ;SWAB,2
000000E0                00C9  2704          .LONG   ^X000000E0                      ;SWAB,3
000000F0                00CD  2705          .LONG   ^X000000F0                      ;SWAB,4
00000DC0                00D1  2706          .LONG   ^X00000DC0                      ;SXT
00008900                00D5  2707          .LONG   ^X00008900                      ;TRAP,1
```

```
00008940  00D9  2708          .LONG   ^X00008940                      ;TRAP,2
00008980  00DD  2709          .LONG   ^X00008980                      ;TRAP,3
000089C0  00E1  2710          .LONG   ^X000089C0                      ;TRAP,4
00000BC0  00E5  2711          .LONG   ^X00000BC0                      ;TST
00008BC0  00E9  2712          .LONG   ^X00008BC0                      ;TSTB
00007800  00ED  2713          .LONG   ^X00007800                      ;XOR,1
00007880  00F1  2714          .LONG   ^X00007880                      ;XOR,2
          00F5  2715
          00F5  2716 END_CPU_DATA
          0118  2717
          0118  2718
          0118  2719 END_TEST_DATA
          0118
          0118        ;-------------------------------------------------------------------------
          0118  2720 ENDTEST
```

```
                        002E        .SBTTL  "TEST   0D6     CM IRD1 Instruction Address Decode Test 1
                        002E  2722 ;*****************************************************************************
                        002E  2723 ;++
                        002E  2724 ;
                        002E  2725 ; FUNCTIONAL DESCRIPTION
                        002E  2726 ;
                        002E  2727 ;       This test makes sure the proper control store addresses are generated
                        002E  2728 ;       for compatiblity mode IRD1s. The test is implemented by using two
                        002E  2729 ;       data tables. The first table consists of generated VAX compatibility
                        002E  2730 ;       mode macro instructions. The second table consists of expected control
                        002E  2731 ;       store addresses for the IRD1.
                        002E  2732 ;
                        002E  2733 ; TEST ALGORITHM
                        002E  2734 ;
                        002E  2735 ;       This test is executed in the following manner:
                        002E  2736 ;
                        002E  2737 ;        1. Initialize the CPU.
                        002E  2738 ;        2. Execute the micro-code to place the CPU into compatibility mode.
                        002E  2739 ;        3. Initialize the WD register with zero.
                        002E  2740 ;        4. Prepare a loop for 256 instructions.
                        002E  2741 ;        5. Load the WD register with a model opcode from the first table
                        002E  2742 ;           indexed by the loop count.
                        002E  2743 ;        6. Execute the micro-code to:
                        002E  2744 ;                1. Load RTEMP1 with the data from the WD register.
                        002E  2745 ;                2. Load the VA with zero.
                        002E  2746 ;                3. Write the model instruction from RTEMP1 to cache.
                        002E  2747 ;                4. Increment the VA for the next address.
                        002E  2748 ;                5. Write zeros into this cache location to prevent cache
                        002E  2749 ;                   parity errors from trapping the IRD1 decode.
                        002E  2750 ;                6. Zero the PC which will initiate instruction prefetch.
                        002E  2751 ;                7. Execute a BUT/IRD1.
                        002E  2752 ;                8. Halt.
                        002E  2753 ;        7. Execute a masked comparison of the control store address trace
                        002E  2754 ;           register with the expected CS address.
                        002E  2755 ;        8. If the two are different signal an error.
                        002E  2756 ;        9. Otherwise continue looping from step 4 to step 8 till all 256
                        002E  2757 ;           instructions have been decoded.
                        002E  2758 ;       10. Execute the micro-code to place the CPU back into native mode.
                        002E  2759 ;       11. Exit.
                        002E  2760 ;
                        002E  2761 ; DATA PATTERNS
                        002E  2762 ;
                        002E  2763 ;       The input data patterns were derived from chapter 4.4 of the CPU
                        002E  2764 ;       micro-sequencer specification dated on 9/25/78. Because operand
                        002E  2765 ;       specifiers are embedded into compatibility mode instructions, certain
                        002E  2766 ;       instructions have more than one opcode decode. Model opcodes were
                        002E  2767 ;       generated by first using Chart 4.4.1B to obtain the base opcodes for
                        002E  2768 ;       each compatibility mode instruction. These opcodes were then enhanced
                        002E  2769 ;       to include all the possible decodes. The enhancements were obtained
                        002E  2770 ;       by changing the operands to specify both register and non-register
                        002E  2771 ;       modes and by using Chart 4.4.1A to modify the operands to get all
                        002E  2772 ;       possible instruction decodes.
                        002E  2773 ;
                        002E  2774 ;       The expected control store addresses where obtained from the
                        002E  2775 ;       Compatibility Mode Ird Rom Definition in the CPU micro-code.
                        002E  2776 ;
                        002E  2777 ;       The following is a table that shows the compatibility mode instruction
```

```
002E  2778 ;       being decoded for a corresponding range of loop counts:
002E  2779 ;
002E  2780 ;       Loop            Instruction             Loop            Instruction
002E  2781 ;
002E  2782 ;       01 - 10         JSR                     94              FSUB
002E  2783 ;       11 - 12         CLR                     95              FMUL
002E  2784 ;       13 - 14         COM                     96              FDIV
002E  2785 ;       15 - 16         INC                     97 - 9E         (unused)
002E  2786 ;       17 - 18         DEC                     9F - A0         SOB
002E  2787 ;       19 - 1A         NEG                     A1 - BC         (reserved)
002E  2788 ;       1B - 1C         ADC                     BD              BR
002E  2789 ;       1D - 1E         SBC                     BE              BNE
002E  2790 ;       1F - 20         TST                     BF              BEQ
002E  2791 ;       21 - 22         ROR                     C0              BGE
002E  2792 ;       23 - 24         ROL                     C1              BLT
002E  2793 ;       25 - 26         ASR                     C2              BGT
002E  2794 ;       27 - 28         ASL                     C3              BLE
002E  2795 ;       29 - 2A         MARK                    C4              BPL
002E  2796 ;       2B - 2C         MFPI                    C5              BMI
002E  2797 ;       2D - 2E         MTPI                    C6              BHI
002E  2798 ;       2F - 30         SXT                     C7              BLOS
002E  2799 ;       31 - 40         (unused)                C8              BVC
002E  2800 ;       41 - 48         EMT                     C9              BVS
002E  2801 ;       49 - 50         TRAP                    CA              BCC, BHIS
002E  2802 ;       51 - 52         CLRB                    CB              BCS, BLO
002E  2803 ;       53 - 54         COMB                    CC              HALT
002E  2804 ;       55 - 56         INCB                    CD              WAIT
002E  2805 ;       57 - 58         DECB                    CE              RTI
002E  2806 ;       59 - 5A         NEGB                    CF              BPT
002E  2807 ;       5B - 5C         ADCB                    D0              IOT
002E  2808 ;       5D - 5E         SBCB                    D1              RESET
002E  2809 ;       5F - 60         TSTB                    D2              RTT
002E  2810 ;       61 - 62         RORB                    D3              (unused)
002E  2811 ;       63 - 64         ROLB                    D4 - D7         MOV
002E  2812 ;       65 - 66         ASRB                    D8 - DB         CMP
002E  2813 ;       67 - 68         ASLB                    DC - DF         BIT
002E  2814 ;       69 - 6A         (unused)                E0 - E3         BIC
002E  2815 ;       6B - 6C         MFPD                    E4 - E7         BIS
002E  2816 ;       6D - 6E         MTPD                    E8 - EB         ADD
002E  2817 ;       6F - 80         (unused)                EC - EF         MOVB
002E  2818 ;       81 - 84         MUL                     F0 - F3         CMPB
002E  2819 ;       85 - 88         DIV                     F4 - F7         BITB
002E  2820 ;       89 - 8C         ASH                     F8 - FB         BICB
002E  2821 ;       8D - 90         ASHC                    FC - FF         BISB
002E  2822 ;       91 - 92         XOR                     100             SUB
002E  2823 ;       93              FADD
002E  2824 ;
002E  2825 ; LOGICAL DESCRIPTION
002E  2826 ;
002E  2827 ;       During an IRD1 in compatibility mode the CCS ROMs are fed addresses
002E  2828 ;       from the Compatibility Mode IRD1 and IRDX ROMs on the DPM. The DPM ROMs
002E  2829 ;       are addressed according to the following scheme:
002E  2830 ;
002E  2831 ;       Source          ROM Address Bits        Comment
002E  2832 ;
002E  2833 ;       CIR <7:0>       <10:3>                  Encoded opcode generated by the
002E  2834 ;                                               IRD gate array.
```

```
002E ;
002E ;       IRD1            <2>                    Distinguishes between the 1st
002E ;                                              operand and the 2nd operand or
002E ;                                              execution flows. 1 during IRD1
002E ;                                              0 otherwise.
002E ;
002E ;       IRD Counter <0> <1>                    Distinguishes between the 2nd
002E ;                                              operand and execution flows.
002E ;                                              0 when decoding OS2, 1 when
002E ;                                              transferring to execution flows
002E ;
002E ;       Register Mode   <0>                    1 if the operand being eval-
002E ;                                              uated specifies register mode.
002E ;                                              Generated by the IRD and PHB
002E ;                                              gate arrays.
002E ;
002E ;       For zero and certain single operand instructions the addresses gener-
002E ;       ated will be execution flows in the micro-code. For the rest, operand
002E ;       specifier evaulation addresses will be generated.
002E ;
002E ; ASSUMPTIONS
002E ;
002E ;       This test assumes the following logic has already been tested; the
002E ;       cache, XB's, the IR and the OSR.
002E ;
002E ; ERROR DESCRIPTION
002E ;
002E ;       Below is a table of possible error symptoms and their most likely
002E ;       causes:
002E ;
002E ;               Symptom                          Probable Cause
002E ;       +-------------------------------+-------------------------------+
002E ;       | The received control store    |                               |
002E ;       | address is bad for one or sev-| One or several bad CM IRD1 and|
002E ;       | eral instructions, or shows   | IRDX ROMs.                    |
002E ;       | certain bits stuck at 1 or 0. |                               |
002E ;       +-------------------------------+-------------------------------+
002E ;       | The received control store    |                               |
002E ;       | address is bad for every      |                               |
002E ;       | instruction or branches to the| A bad IRD gate array.         |
002E ;       | CM.BAD.IRD address in the     |                               |
002E ;       | micro-code.                   |                               |
002E ;       +-------------------------------+-------------------------------+
002E ;       | Register mode fails to assert |                               |
002E ;       | causing the wrong address to  | A bad IRD or PHB gate array.  |
002E ;       | be generated.                 |                               |
002E ;       +-------------------------------+-------------------------------+
002E ;
002E ;       On error this test will call out the IRD and PHB gate arrays and will
002E ;       also type out the following data:
002E ;
002E ;       EXPECTED CONTROL STORE ADDRESS
002E ;       RECEIVED CONTROL STORE ADDRESS
002E ;       LOOP COUNT
002E ;       DATA RESERVED FOR INSTRUCTION
002E ;       OPCODE OF THE INSTRUCTION BEING DECODED
002E ;
```

```
                        002E  2892 ;--
                        002E  2893 ;*****************************************************************************
                        002E  2894 ;/////////////////////////////////////////////////////////////////////////////
                        002E  2895
                        002E  2896          INITIALIZE                              ; INITIALIZE THE CPU
                        0030  2897          FETCH           0                       ; PLACE CPU INTO COMP. MODE
                        0035  2898          BRSTCLK         3                       ;  ...
                        0039  2899          LOADREG         WDREG,1$,,L             ; INITIALIZE THE WD REG. WITH 0
                        0041  2900          EXPUTRAP                                ; EXPECT NON DCS ADDRESSES
                        0043  2901          LOOP            I,1,256                 ; LOOP FOR 256 INSTRUCTIONS
                        004A  2902          LOADREG         WDREG,2$,I,W            ; LOAD AN INST. INTO WD REG.
                        0052  2903          ERRLOOP                                 ; ERROR LOOP STARTS HERE
                        0054  2904          FETCH           4                       ; LOAD INSTRUCTION AND DO AN
                        0059  2905          BRSTCLK         <^X0E>                  ; IRD1
                        005D  2906          CMPREGMSK       CSTRP,3$,I,4$,,W        ; IS THE CS ADDRESS CORRECT?
                        0069  2907          IFERROR         2,<<IRD><PHB>>          ; IF NOT SIGNAL AN ERROR
                        0070  2908          ENDLOOP         I                       ; CONTINUE TILL LOOP IS DONE
                        0073  2909          FETCH           <^X0F>                  ; PUT CPU INTO NATIVE MODE
                        0078  2910          BRSTCLK         <^X12>                  ;  ...
                        007C  2911          SKIP                                    ; END OF TEST
                        0083  2912
                        0083  2913 BEGIN_TEST_DATA
                        0083  2914          ;-------------------------------------------------------------------
                        0083  2915 1$:      .LONG   0       ; LONGWORD TO INITIALIZE THE WD REGISTER
                        0087  2916 ;
                        0087  2917 ; Table of test opcodes.
                        0087  2918 ;
                        0087  2919 ;       The numbers to the right of the semicolons indicates the range
                        0087  2920 ;       of the loop count for those patterns.
                        0087  2921 ;
08C0 0888 0880 0848 0840 0808 0800      0087  2922 2$:      .WORD   ^X0800,^X0808,^X0840,^X0848,^X0880,^X0888,^X08C0 ; 01 - 07
0988 0980 0948 0940 0908 0900 08C8      0095  2923          .WORD   ^X08C8,^X0900,^X0908,^X0940,^X0948,^X0980,^X0988 ; 08 - 0E
0A80 0A48 0A40 0A08 0A00 09C8 09C0      00A3  2924          .WORD   ^X09C0,^X09C8,^X0A00,^X0A08,^X0A40,^X0A48,^X0A80 ; 0F - 15
0B48 0B40 0B08 0B00 0AC8 0AC0 0A88      00B1  2925          .WORD   ^X0A88,^X0AC0,^X0AC8,^X0B00,^X0B08,^X0B40,^X0B48 ; 16 - 1C
0C40 0C08 0C00 0BC8 0BC0 0B88 0B80      00BF  2926          .WORD   ^X0B80,^X0B88,^X0BC0,^X0BC8,^X0C00,^X0C08,^X0C40 ; 1D - 23
0D08 0D00 0CC8 0CC0 0C88 0C80 0C48      00CD  2927          .WORD   ^X0C48,^X0C80,^X0C88,^X0CC0,^X0CC8,^X0D00,^X0D08 ; 24 - 2A
0E00 0DC8 0DC0 0D88 0D80 0D48 0D40      00DB  2928          .WORD   ^X0D40,^X0D48,^X0D80,^X0D88,^X0DC0,^X0DC8,^X0E00 ; 2B - 31
0EC8 0EC0 0E88 0E80 0E48 0E40 0E08      00E9  2929          .WORD   ^X0E08,^X0E40,^X0E48,^X0E80,^X0E88,^X0EC0,^X0EC8 ; 32 - 38
0FC0 0F88 0F80 0F48 0F40 0F08 0F00      00F7  2930          .WORD   ^X0F00,^X0F08,^X0F40,^X0F48,^X0F80,^X0F88,^X0FC0 ; 39 - 3F
8888 8880 8848 8840 8808 8800 0FC8      0105  2931          .WORD   ^X0FC8,^X8800,^X8808,^X8840,^X8848,^X8880,^X8888 ; 40 - 46
8980 8948 8940 8908 8900 88C8 88C0      0113  2932          .WORD   ^X88C0,^X88C8,^X8900,^X8908,^X8940,^X8948,^X8980 ; 47 - 4D
8A48 8A40 8A08 8A00 89C8 89C0 8988      0121  2933          .WORD   ^X8988,^X89C0,^X89C8,^X8A00,^X8A08,^X8A40,^X8A48 ; 4E - 54
8B40 8B08 8B00 8AC8 8AC0 8A88 8A80      012F  2934          .WORD   ^X8A80,^X8A88,^X8AC0,^X8AC8,^X8B00,^X8B08,^X8B40 ; 55 - 5B
8C08 8C00 8BC8 8BC0 8B88 8B80 8B48      013D  2935          .WORD   ^X8B48,^X8B80,^X8B88,^X8BC0,^X8BC8,^X8C00,^X8C08 ; 5C - 62
8D00 8CC8 8CC0 8C88 8C80 8C48 8C40      014B  2936          .WORD   ^X8C40,^X8C48,^X8C80,^X8C88,^X8CC0,^X8CC8,^X8D00 ; 63 - 69
8DC8 8DC0 8D88 8D80 8D48 8D40 8D08      0159  2937          .WORD   ^X8D08,^X8D40,^X8D48,^X8D80,^X8D88,^X8DC0,^X8DC8 ; 6A - 70
8EC0 8E88 8E80 8E48 8E40 8E08 8E00      0167  2938          .WORD   ^X8E00,^X8E08,^X8E40,^X8E48,^X8E80,^X8E88,^X8EC0 ; 71 - 77
8F88 8F80 8F48 8F40 8F08 8F00 8EC8      0175  2939          .WORD   ^X8EC8,^X8F00,^X8F08,^X8F40,^X8F48,^X8F80,^X8F88 ; 78 - 7E
7200 7088 7080 7008 7000 8FC8 8FC0      0183  2940          .WORD   ^X8FC0,^X8FC8,^X7000,^X7008,^X7080,^X7088,^X7200 ; 7F - 85
7488 7480 7408 7400 7288 7280 7208      0191  2941          .WORD   ^X7208,^X7280,^X7288,^X7400,^X7408,^X7480,^X7488 ; 86 - 8C
7A00 7880 7800 7688 7680 7608 7600      019F  2942          .WORD   ^X7600,^X7608,^X7680,^X7688,^X7800,^X7880,^X7A00 ; 8D - 93
7AA8 7AA0 7A28 7A20 7A18 7A10 7A08      01AD  2943          .WORD   ^X7A08,^X7A10,^X7A18,^X7A20,^X7A28,^X7AA0,^X7AA8 ; 94 - 9A
F000 7E80 7E00 7CA8 7CA0 7C28 7C20      01BB  2944          .WORD   ^X7C20,^X7C28,^X7CA0,^X7CA8,^X7E00,^X7E80,^XF000 ; 9B - A1
F288 F280 F208 F200 F088 F080 F008      01C9  2945          .WORD   ^XF008,^XF080,^XF088,^XF200,^XF208,^XF280,^XF288 ; A2 - A8
F800 F688 F680 F608 F600 F480 F400      01D7  2946          .WORD   ^XF400,^XF480,^XF600,^XF608,^XF680,^XF688,^XF800 ; A9 - AF
```

```
 FA88 FA80 FA08 FA00 F888 F880 F808  01E5  2947          .WORD   ^XF808,^XF880,^XF888,^XFA00,^XFA08,^XFA80,^XFA88 ; B0 - B6
 0100 FE80 FE00 FC88 FC80 FC08 FC00  01F3  2948          .WORD   ^XFC00,^XFC08,^XFC80,^XFC88,^XFE00,^XFE80,^X0100 ; B7 - BD
 8000 0700 0600 0500 0400 0300 0200  0201  2949          .WORD   ^X0200,^X0300,^X0400,^X0500,^X0600,^X0700,^X8000 ; BE - C4
 8700 8600 8500 8400 8300 8200 8100  020F  2950          .WORD   ^X8100,^X8200,^X8300,^X8400,^X8500,^X8600,^X8700 ; C5 - CB
 0006 0005 0004 0003 0002 0001 0000  021D  2951          .WORD   ^X0000,^X0001,^X0002,^X0003,^X0004,^X0005,^X0006 ; CC - D2
 2200 2000 1280 1200 1080 1000 0007  022B  2952          .WORD   ^X0007,^X1000,^X1080,^X1200,^X1280,^X2000,^X2200 ; D3 - D9
 4000 3280 3080 3200 3000 2280 2080  0239  2953          .WORD   ^X2080,^X2280,^X3000,^X3200,^X3080,^X3280,^X4000 ; DA - E0
 5280 5200 5080 5000 4280 4200 4080  0247  2954          .WORD   ^X4080,^X4200,^X4280,^X5000,^X5080,^X5200,^X5280 ; E1 - E7
 9200 9080 9000 6280 6200 6080 6000  0255  2955          .WORD   ^X6000,^X6080,^X6200,^X6280,^X9000,^X9080,^X9200 ; E8 - EE
 B080 B000 A280 A200 A080 A000 9280  0263  2956          .WORD   ^X9280,^XA000,^XA080,^XA200,^XA280,^XB000,^XB080 ; EF - F5
 D000 C280 C200 C080 C000 B280 B200  0271  2957          .WORD   ^XB200,^XB280,^XC000,^XC080,^XC200,^XC280,^XD000 ; F6 - FC
           E000 D280 D200 D080       027F  2958          .WORD   ^XD080,^XD200,^XD280,^XE000                      ; FD - 100
                                     0287  2959 ;
                                     0287  2960 ; Table of control store addresses.
                                     0287  2961 ;
                                     0287  2962 ;       The numbers to the right of the semicolons indicates the range
                                     0287  2963 ;       of the loop count for those patterns.
                                     0287  2964 ;
 02F0 02F2 02F0 02F2 02F0 02F2 02F0  0287  2965 3$:      .WORD   ^X02F0,^X02F2,^X02F0,^X02F2,^X02F0,^X02F2,^X02F0 ; 01 - 07
 02F2 02F0 02F2 02F0 02F2 02F0 02F2  0295  2966          .WORD   ^X02F2,^X02F0,^X02F2,^X02F0,^X02F2,^X02F0,^X02F2 ; 08 - 0E
 001B 02E2 006B 02F2 0013 02F2 02F0  02A3  2967          .WORD   ^X02F0,^X02F2,^X0013,^X02F2,^X006B,^X02E2,^X001B ; 0F - 15
 02E2 008B 02E2 0061 02E2 0017 02E2  02B1  2968          .WORD   ^X02E2,^X0017,^X02E2,^X0061,^X02E2,^X008B,^X02E2 ; 16 - 1C
 00A5 02E2 00AB 02D2 0067 02E2 0091  02BF  2969          .WORD   ^X0091,^X02E2,^X0067,^X02D2,^X00AB,^X02E2,^X00A5 ; 1D - 23
 010F 010F 02E2 0077 02E2 0071 02E2  02CD  2970          .WORD   ^X02E2,^X0071,^X02E2,^X0077,^X02E2,^X010F,^X010F ; 24 - 2A
 010F 02F2 03D7 02F2 03E9 02D2 00D7  02DB  2971          .WORD   ^X00D7,^X02D2,^X03E9,^X02F2,^X03D7,^X02F2,^X010F ; 2B - 31
 010F 010F 010F 010F 010F 010F 010F  02E9  2972          .WORD   ^X010F,^X010F,^X010F,^X010F,^X010F,^X010F,^X010F ; 32 - 38
 010F 010F 010F 010F 010F 010F 010F  02F7  2973          .WORD   ^X010F,^X010F,^X010F,^X010F,^X010F,^X010F,^X010F ; 39 - 3F
 011F 011F 011F 011F 011F 011F 010F  0305  2974          .WORD   ^X010F,^X011F,^X011F,^X011F,^X011F,^X011F,^X011F ; 40 - 46
 012F 012F 012F 012F 012F 011F 011F  0313  2975          .WORD   ^X011F,^X011F,^X012F,^X012F,^X012F,^X012F,^X012F ; 47 - 4D
 02E2 006B 02F2 0013 012F 012F 012F  0321  2976          .WORD   ^X012F,^X012F,^X012F,^X0013,^X02F2,^X006B,^X02E2 ; 4E - 54
 008B 02E2 0061 02E2 0017 02E2 001B  032F  2977          .WORD   ^X001B,^X02E2,^X0017,^X02E2,^X0061,^X02E2,^X008B ; 55 - 5B
 02E2 00B1 02D2 0067 02E2 0091 02E2  033D  2978          .WORD   ^X02E2,^X0091,^X02E2,^X0067,^X02D2,^X00B1,^X02E2 ; 5C - 62
 010F 02E2 0077 02E2 0071 02E2 00A5  034B  2979          .WORD   ^X00A5,^X02E2,^X0071,^X02E2,^X0077,^X02E2,^X010F ; 63 - 69
 010F 010F 02F2 03E9 02D2 00D7 010F  0359  2980          .WORD   ^X010F,^X00D7,^X02D2,^X03E9,^X02F2,^X010F,^X010F ; 6A - 70
 010F 010F 010F 010F 010F 010F 010F  0367  2981          .WORD   ^X010F,^X010F,^X010F,^X010F,^X010F,^X010F,^X010F ; 71 - 77
 010F 010F 010F 010F 010F 010F 010F  0375  2982          .WORD   ^X010F,^X010F,^X010F,^X010F,^X010F,^X010F,^X010F ; 78 - 7E
 02D0 02D2 02D0 02D2 02D0 010F 010F  0383  2983          .WORD   ^X010F,^X010F,^X02D0,^X02D2,^X02D0,^X02D2,^X02D0 ; 7F - 85
 02D2 02D0 02D2 02D0 02D2 02D0 02D2  0391  2984          .WORD   ^X02D2,^X02D0,^X02D2,^X02D0,^X02D2,^X02D0,^X02D2 ; 86 - 8C
 010F 013F 013F 02D2 02D0 02D2 02D0  039F  2985          .WORD   ^X02D0,^X02D2,^X02D0,^X02D2,^X013F,^X013F,^X010F ; 8D - 93
 010F 010F 010F 010F 010F 010F 010F  03AD  2986          .WORD   ^X010F,^X010F,^X010F,^X010F,^X010F,^X010F,^X010F ; 94 - 9A
 010F 015F 015F 010F 010F 010F 010F  03BB  2987          .WORD   ^X010F,^X010F,^X010F,^X010F,^X015F,^X015F,^X010F ; 9B - A1
 010F 010F 010F 010F 010F 010F 010F  03C9  2988          .WORD   ^X010F,^X010F,^X010F,^X010F,^X010F,^X010F,^X010F ; A2 - A8
 010F 010F 010F 010F 010F 010F 010F  03D7  2989          .WORD   ^X010F,^X010F,^X010F,^X010F,^X010F,^X010F,^X010F ; A9 - AF
 010F 010F 010F 010F 010F 010F 010F  03E5  2990          .WORD   ^X010F,^X010F,^X010F,^X010F,^X010F,^X010F,^X010F ; B0 - B6
 014E 010F 010F 010F 010F 010F 010F  03F3  2991          .WORD   ^X010F,^X010F,^X010F,^X010F,^X010F,^X010F,^X014E ; B7 - BD
 015E 015E 015E 015E 015E 015E 015E  0401  2992          .WORD   ^X015E,^X015E,^X015E,^X015E,^X015E,^X015E,^X015E ; BE - C4
 015E 015E 015E 015E 015E 015E 015E  040F  2993          .WORD   ^X015E,^X015E,^X015E,^X015E,^X015E,^X015E,^X015E ; C5 - CB
 017F 010F 001E 000E 017F 010F 010F  041D  2994          .WORD   ^X010F,^X010F,^X017F,^X000E,^X001E,^X010F,^X017F ; CC - D2
 02D2 02D0 02D2 02D2 02D0 02D0 010F  042B  2995          .WORD   ^X010F,^X02D0,^X02D0,^X02D2,^X02D2,^X02D0,^X02D2 ; D3 - D9
 02D0 02D2 02D0 02D2 02D0 02D2 02D0  0439  2996          .WORD   ^X02D0,^X02D2,^X02D0,^X02D2,^X02D0,^X02D2,^X02D0 ; DA - E0
 02D2 02D2 02D0 02D0 02D2 02D2 02D0  0447  2997          .WORD   ^X02D0,^X02D2,^X02D2,^X02D0,^X02D0,^X02D2,^X02D2 ; E1 - E7
 02D2 02D0 02D0 02D2 02D2 02D0 02D0  0455  2998          .WORD   ^X02D0,^X02D0,^X02D2,^X02D2,^X02D0,^X02D0,^X02D2 ; E8 - EE
 02D0 02D0 02D2 02D2 02D0 02D0 02D2  0463  2999          .WORD   ^X02D2,^X02D0,^X02D0,^X02D2,^X02D2,^X02D0,^X02D0 ; EF - F5
 02D0 02D2 02D2 02D0 02D0 02D2 02D2  0471  3000          .WORD   ^X02D2,^X02D2,^X02D0,^X02D0,^X02D2,^X02D2,^X02D0 ; F6 - FC
           02D0 02D2 02D2 02D0       047F  3001          .WORD   ^X02D0,^X02D2,^X02D2,^X02D0                      ; FD - 100
                                     0487  3002
                                3FFF 0487  3003 4$:      .WORD   ^X3FFF  ; MASK FOR ADDRESS COMPARISON
```

```
                                0489  3004
                                0489  3005 BEGIN_CPU_DATA
                                0002  3006
                                0002  3007 DCS_DATA
                                0001  3008
U 00, 7700,C800,0364,0300,0470,1801   0001  3009 ;% 00   NOP                                 ; PLACES CPU INTO COMP. MODE
U 01, 7700,3800,3FFF,FFFF,8470,1802   000C  3013 ;% 01   LONLIT_[80000000]
U 02, 7700,C800,5BE4,03D4,0070,1803   0017  3017 ;% 02   WB_R[LONLIT],CCPSL/PSL_WB.CCBR_ALUS
U 03, 7300,C800,0364,0300,0470,1804   0022  3021 ;% 03   STOP.CLOCK
                                002D  3025
U 04, 7700,4800,0364,0300,0470,1805   002D  3026 ;% 04   NOP                                 ; GET THE CLOCK UP TO SPEED
U 05, 5700,4840,0364,0304,0470,1806   0038  3030 ;% 05   R[TEMP1]_RDM                         ; PUT MODEL OPCODE INTO RTEMP1
U 06, 7700,C800,5BE4,0314,4A70,1807   0043  3034 ;% 06   VA_R[TEMP5]                         ; ZERO THE VA
U 07, 7701,C800,5BE4,0204,5590,1808   004E  3038 ;% 07   WRITE.LONG R[TEMP1]                 ; WRITE OPCODE TO CACHE FOR XB0
U 08, 7701,4800,5BE4,0204,55F0,1809   0059  3042 ;% 08   WDR_UNROT(R[TEMP1]),BUS/PRB.RD,SIZE[LONG]
                                0064  3046                                                ; HOLD FOR SECOND CYCLE
U 09, 7700,C800,0364,0300,4470,180A   0064  3047 ;% 09   VA_VA+4                           ; INCREMENT THE VA
U 0A, 7701,8800,5BE4,0214,5590,180B   006F  3051 ;% 0A   WRITE.LONG R[TEMP5]                 ; WRITE ZERO TO CACHE TO VALIDATE
U 0B, 7701,0800,5BE4,0214,55F0,180C   007A  3055 ;% 0B   WDR_UNROT(R[TEMP5]),BUS/PRB.RD,SIZE[LONG]
                                0085  3059                                                ; LOCATION FOR XB1
U 0C, 7700,4800,5BE4,0314,4870,180D   0085  3060 ;% 0C   PC_R[TEMP5]                         ; INITIATE PREFETCH
U 0D, 7700,8800,0364,1300,0470,180E   0090  3064 ;% 0D   BUT/IRD1                            ; DO AN IRD1 DECODE
U 0E, 7300,4800,0364,0300,0470,180F   009B  3068 ;% 0E   STOP.CLOCK                          ; HALT
                                00A6  3072
U 0F, 7700,C800,0364,0300,0470,1810   00A6  3073 ;% 0F   NOP                                 ; PLACE CPU BACK INTO NATIVE MODE
U 10, 7700,F800,7FFF,FFFF,8470,1811   00B1  3077 ;% 10   LONLIT_[0]
U 11, 7700,C800,5BE4,03D4,0070,1812   00BC  3081 ;% 11   WB_R[LONLIT],CCPSL/PSL_WB.CCBR_ALUS
U 12, 7300,C800,0364,0300,0470,1813   00C7  3085 ;% 12   STOP.CLOCK
                                00D2  3089
                                00D2  3090 RTEMP_DATA
                                0001  3091
                     00000000   0001  3092          .LONG   0       ; RTEMP0 = reserved for instruction use
                     00000000   0005  3093          .LONG   0       ; RTEMP1 = INSTRUCTION OPCODE
                     00000000   0009  3094          .LONG   0       ; RTEMP2 = unused
                     00000000   000D  3095          .LONG   0       ; RTEMP3 = unused
                     00000000   0011  3096          .LONG   0       ; RTEMP4 = unused
                     00000000   0015  3097          .LONG   0       ; RTEMP5 = INSTRUCTION ADDRESS
                                0019  3098
                                0019  3099 END_CPU_DATA
                                0489  3100
                                0489  3101 END_TEST_DATA
                                0489
                                0489        ;-----------------------------------------------------------------------
                                0489  3102 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
                        002E        .SBTTL  "TEST   0D7     CM IRD1 Instruction Address Decode Test 2
                        002E  3104 ;*******************************************************************************
                        002E  3105 ;++
                        002E  3106 ;
                        002E  3107 ; FUNCTIONAL DESCRIPTION
                        002E  3108 ;
                        002E  3109 ;       This test makes sure the proper control store addresses are generated
                        002E  3110 ;       for compatiblity mode IRD1s. The test is implemented by using two
                        002E  3111 ;       data tables. The first table consists of assembled VAX compatibility
                        002E  3112 ;       mode macro instructions. The second table consists of expected control
                        002E  3113 ;       store addresses for the IRD1. This test executes those instructions
                        002E  3114 ;       which could not be decoded in test one due to RDM memory contraints.
                        002E  3115 ;
                        002E  3116 ; TEST ALGORITHM
                        002E  3117 ;
                        002E  3118 ;       This test is executed in the following manner:
                        002E  3119 ;
                        002E  3120 ;        1. Initialize the CPU.
                        002E  3121 ;        2. Execute the micro-code to place the CPU into compatibility mode.
                        002E  3122 ;        3. Initialize the WD register with zero.
                        002E  3123 ;        4. Prepare a loop for 28 instructions.
                        002E  3124 ;        5. Load the WD register with a model opcode from the first table
                        002E  3125 ;           indexed by the loop count.
                        002E  3126 ;        6. Execute the micro-code to:
                        002E  3127 ;                1. Load RTEMP1 with the data from the WD register.
                        002E  3128 ;                2. Load the VA with zero.
                        002E  3129 ;                3. Write the model instruction from RTEMP1 to cache.
                        002E  3130 ;                4. Increment the VA for the next address.
                        002E  3131 ;                5. Write zeros into this cache location to prevent cache
                        002E  3132 ;                   parity errors from trapping the IRD1 decode.
                        002E  3133 ;                6. Zero the PC which will initiate instruction prefetch.
                        002E  3134 ;                7. Execute a BUT/IRD1.
                        002E  3135 ;                8. Halt.
                        002E  3136 ;        7. Execute a masked comparison of the control store address trace
                        002E  3137 ;           register with the expected CS address.
                        002E  3138 ;        8. If the two are different signal an error.
                        002E  3139 ;        9. Otherwise continue looping from step 4 to step 8 till finished.
                        002E  3140 ;       10. Execute the micro-code to place the CPU back into native mode.
                        002E  3141 ;       11. Exit.
                        002E  3142 ;
                        002E  3143 ; DATA PATTERNS
                        002E  3144 ;
                        002E  3145 ;       The input data patterns were derived from chapter 4.4 of the CPU
                        002E  3146 ;       micro-sequencer specification dated on 9/25/78. Because operand
                        002E  3147 ;       specifiers are embedded into compatibility mode instructions, certain
                        002E  3148 ;       instructions have more than one opcode decode. Model opcodes were
                        002E  3149 ;       generated by first using Chart 4.4.1B to obtain the base opcodes for
                        002E  3150 ;       each compatibility mode instruction. These opcodes were then enhanced
                        002E  3151 ;       to include all the possible decodes. The enhancements were obtained
                        002E  3152 ;       by changing the operands to specify both register and non-register
                        002E  3153 ;       modes and by using Chart 4.4.1A to modify the operands to get all
                        002E  3154 ;       possible instruction decodes.
                        002E  3155 ;
                        002E  3156 ;       The expected control store addresses where obtained from the
                        002E  3157 ;       Compatibility Mode Ird Rom Definition in the CPU micro-code.
                        002E  3158 ;
                        002E  3159 ;       The following is a table that shows the instruction being decoded for
```

```
002E  3160 ;       a range of loop counts:
002E  3161 ;
002E  3162 ;       Loop            Instruction
002E  3163 ;
002E  3164 ;       01 - 03         SUB
002E  3165 ;       04 - 0B         JMP
002E  3166 ;       0C              RTS
002E  3167 ;       0D - 0E         (unused)
002E  3168 ;       0F              SPL
002E  3169 ;       10              NOP
002E  3170 ;       11 - 14         (reserved)
002E  3171 ;       15 - 1C         SWAB
002E  3172 ;
002E  3173 ; LOGICAL DESCRIPTION
002E  3174 ;
002E  3175 ;       During an IRD1 in compatibility mode the CCS ROMs are fed addresses
002E  3176 ;       from the Compatibility Mode IRD1 and IRDX ROMs on the DPM. The DPM ROMs
002E  3177 ;       are addressed according to the following scheme:
002E  3178 ;
002E  3179 ;       Source          ROM Address Bits        Comment
002E  3180 ;
002E  3181 ;       CIR <7:0>       <10:3>                  Encoded opcode generated by the
002E  3182 ;                                               IRD gate array.
002E  3183 ;
002E  3184 ;       IRD1            <2>                     Distinguishes between the 1st
002E  3185 ;                                               operand and the 2nd operand or
002E  3186 ;                                               execution flows. 1 during IRD1
002E  3187 ;                                               0 otherwise.
002E  3188 ;
002E  3189 ;       IRD Counter <0> <1>                     Distinguishes between the 2nd
002E  3190 ;                                               operand and execution flows.
002E  3191 ;                                               0 when decoding OS2, 1 when
002E  3192 ;                                               transferring to execution flows
002E  3193 ;
002E  3194 ;       Register Mode   <0>                     1 if the operand being eval-
002E  3195 ;                                               uated specifies register mode.
002E  3196 ;                                               Generated by the IRD and PHB
002E  3197 ;                                               gate arrays.
002E  3198 ;
002E  3199 ;       For zero and certain single operand instructions the addresses gener-
002E  3200 ;       ated will be execution flows in the micro-code. For the rest, operand
002E  3201 ;       specifier evaulation addresses will be generated.
002E  3202 ;
002E  3203 ; ASSUMPTIONS
002E  3204 ;
002E  3205 ;       This test assumes the following logic has already been tested; the
002E  3206 ;       cache, XB's, the IR and the OSR.
002E  3207 ;
002E  3208 ; ERROR DESCRIPTION
002E  3209 ;
002E  3210 ;       Below is a table of possible error symptoms and their most likely
002E  3211 ;       causes:
002E  3212 ;
002E  3213 ;              Symptom                          Probable Cause
002E  3214 ;       +-------------------------------+-------------------------------+
002E  3215 ;       | The received control store    |                               |
002E  3216 ;       | address is bad for one or sev-| One or several bad CM IRD1 and|
```

```
                                002E  ;       | eral instructions, or shows   | IRDX ROMs.                    |
                                002E  ;       | certain bits stuck at 1 or 0. |                               |
                                002E  ;       +-------------------------------+-------------------------------+
                                002E  ;       | The received control store    |                               |
                                002E  ;       | address is bad for every      |                               |
                                002E  ;       | instruction or branches to the| A bad IRD gate array.         |
                                002E  ;       | CM.BAD.IRD address in the     |                               |
                                002E  ;       | micro-code.                   |                               |
                                002E  ;       +-------------------------------+-------------------------------+
                                002E  ;       | Register mode fails to assert |                               |
                                002E  ;       | causing the wrong address to  | A bad IRD or PHB gate array.  |
                                002E  ;       | be generated.                 |                               |
                                002E  ;       +-------------------------------+-------------------------------+
                                002E  ;
                                002E  ;       On error this test will call out the IRD and PHB gate arrays and will
                                002E  ;       also type out the following data:
                                002E  ;
                                002E  ;       EXPECTED CONTROL STORE ADDRESS
                                002E  ;       RECEIVED CONTROL STORE ADDRESS
                                002E  ;       LOOP COUNT
                                002E  ;       DATA RESERVED FOR INSTRUCTION
                                002E  ;       OPCODE OF THE INSTRUCTION BEING DECODED
                                002E  ;
                                002E  ;--
                                002E  ;*******************************************************************************
                                002E  ;///////////////////////////////////////////////////////////////////////////////
                                002E
                                002E          INITIALIZE                                  ; INITIALIZE THE CPU
                                0030          FETCH           0                           ; PLACE CPU INTO COMP. MODE
                                0035          BRSTCLK         3                           ; ...
                                0039          LOADREG         WDREG,1$,,L                 ; INITIALIZE THE WD REG. WITH 0
                                0041          EXPUTRAP                                    ; EXPECT NON-DCS ADDRESS
                                0043          LOOP            I,1,28                      ; LOOP FOR 28 OPCODES
                                004A          LOADREG         WDREG,2$,I,W                ; LOAD MODEL OPCODE
                                0052          ERRLOOP                                     ; ERROR LOOP STARTS HERE
                                0054          FETCH           4                           ; EXECUTE AN IRD1
                                0059          BRSTCLK         <^X0E>,INH                  ; ...
                                005D          CMPREGMSK       CSTRP,3$,I,4$,,W            ; IS THE CS ADDRESS CORRECT?
                                0069          IFERROR         2,<<IRD><PHB>>              ; IF NOT SIGNAL AN ERROR
                                0070          ENDLOOP         I                           ; CONTINUE FOR REST
                                0073          FETCH           <^X0F>                      ; PUT CPU INTO NATIVE MODE
                                0078          BRSTCLK         <^X12>                      ; ...
                                007C          SKIP                                        ; END OF TEST
                                0083
                                0083  BEGIN_TEST_DATA
                                0083
                                0083      ;-----------------------------------------------------------------------
                                0083
                        00000000  0083  1$:     .LONG   0       ; ZERO TO INITIALIZE THE WD REGISTER
                                0087  ;
                                0087  ; Table of test opcodes.
                                0087  ;
                                0087  ;       The numbers to the right of the semicolons indicates the range
                                0087  ;       of the loop count for those patterns.
                                0087  ;
0058 0050 0048 0040 E280 E200 E080  0087  2$:     .WORD   ^XE080,^XE200,^XE280,^X0040,^X0048,^X0050,^X0058 ; 01 - 07
0090 0088 0080 0078 0070 0068 0060  0095          .WORD   ^X0060,^X0068,^X0070,^X0078,^X0080,^X0088,^X0090 ; 08 - 0E
```

```
 00C0 00B8 00B0 00A9 00A1 00A0 0098   00A3  3272          .WORD   ^X0098,^X00A0,^X00A1,^X00A9,^X00B0,^X00B8,^X00C0 ; 0F - 15
 00F8 00F0 00E8 00E0 00D8 00D0 00C8   00B1  3273          .WORD   ^X00C8,^X00D0,^X00D8,^X00E0,^X00E8,^X00F0,^X00F8 ; 16 - 1C
                                      00BF  3274 ;
                                      00BF  3275 ; Table of control store addresses.
                                      00BF  3276 ;
                                      00BF  3277 ;       The numbers to the right of the semicolons indicates the range
                                      00BF  3278 ;       of the loop count for those patterns.
                                      00BF  3279 ;
 02F6 02F4 02F2 00D1 02D2 02D2 02D0   00BF  3280 3$:      .WORD   ^X02D0,^X02D2,^X02D2,^X00D1,^X02F2,^X02F4,^X02F6 ; 01 - 07
 010F 014F 014F 02FE 02FC 02FA 02F8   00CD  3281          .WORD   ^X02F8,^X02FA,^X02FC,^X02FE,^X014F,^X014F,^X010F ; 08 - 0E
 00BB 016F 016F 016F 016F 016F 010F   00DB  3282          .WORD   ^X010F,^X016F,^X016F,^X016F,^X016F,^X016F,^X00BB ; 0F - 15
 02EE 02EC 02EA 02E8 02E6 02E4 02E2   00E9  3283          .WORD   ^X02E2,^X02E4,^X02E6,^X02E8,^X02EA,^X02EC,^X02EE ; 16 - 1C
                                      00F7  3284
                                3FFF  00F7  3285 4$:      .WORD   ^X3FFF          ; MASK FOR ADDRESS COMPARISON
                                      00F9  3286
                                      00F9  3287 BEGIN_CPU_DATA
                                      0002  3288
                                      0002  3289 DCS_DATA
                                      0001  3290
U 00, 7700,C800,0364,0300,0470,1801   0001  3291 ;% 00   NOP                             ; Places CPU into comp.mode
U 01, 7700,3800,3FFF,FFFF,8470,1802   000C  3295 ;% 01   LONLIT_[80000000]
U 02, 7700,C800,5BE4,03D4,0070,1803   0017  3299 ;% 02   WB_R[LONLIT],CCPSL/PSL_WB.CCBR_ALUS
U 03, 7300,C800,0364,0300,0470,1804   0022  3303 ;% 03   STOP.CLOCK
                                      002D  3307
U 04, 7700,4800,0364,0300,0470,1805   002D  3308 ;% 04   NOP                             ; GET THE CLOCK UP TO SPEED
U 05, 5700,4840,0364,0304,0470,1806   0038  3312 ;% 05   R[TEMP1]_RDM                     ; PUT MODEL OPCODE INTO RTEMP1
U 06, 7700,C800,5BE4,0314,4A70,1807   0043  3316 ;% 06   VA_R[TEMP5]                     ; ZERO THE VA
U 07, 7701,C800,5BE4,0204,5590,1808   004E  3320 ;% 07   WRITE.LONG R[TEMP1]            ; WRITE OPCODE TO CACHE FOR XB0
U 08, 7701,4800,5BE4,0204,55F0,1809   0059  3324 ;% 08   WDR_UNROT(R[TEMP1]),BUS/PRB.RD,SIZE[LONG]
                                      0064  3328                                      ; HOLD FOR SECOND CYCLE
U 09, 7700,C800,0364,0300,4470,180A   0064  3329 ;% 09   VA_VA+4                        ; INCREMENT THE VA
U 0A, 7701,8800,5BE4,0214,5590,180B   006F  3333 ;% 0A   WRITE.LONG R[TEMP5]            ; WRITE ZERO TO CACHE TO VALIDATE
U 0B, 7701,0800,5BE4,0214,55F0,180C   007A  3337 ;% 0B   WDR_UNROT(R[TEMP5]),BUS/PRB.RD,SIZE[LONG]
                                      0085  3341                                      ; LOCATION FOR XB1
U 0C, 7700,4800,5BE4,0314,4870,180D   0085  3342 ;% 0C   PC_R[TEMP5]                     ; INITIATE PREFETCH
U 0D, 7700,8800,0364,1300,0470,180E   0090  3346 ;% 0D   BUT/IRD1                       ; DO AN IRD1 DECODE
U 0E, 7300,4800,0364,0300,0470,180F   009B  3350 ;% 0E   STOP.CLOCK                     ; HALT
                                      00A6  3354
U 0F, 7700,C800,0364,0300,0470,1810   00A6  3355 ;% 0F   NOP                             ; PUT THE CPU BACK INTO NATIVE MODE
U 10, 7700,F800,7FFF,FFFF,8470,1811   00B1  3359 ;% 10   LONLIT_[0]
U 11, 7700,C800,5BE4,03D4,0070,1812   00BC  3363 ;% 11   WB_R[LONLIT],CCPSL/PSL_WB.CCBR_ALUS
U 12, 7300,C800,0364,0300,0470,1813   00C7  3367 ;% 12   STOP.CLOCK
                                      00D2  3371
                                      00D2  3372 RTEMP_DATA
                                      0001  3373
                          00000000    0001  3374          .LONG   0               ; RTEMP0 = reserved for instruction use
                          00000000    0005  3375          .LONG   0               ; RTEMP1 = INSTRUCTION OPCODE
                          00000000    0009  3376          .LONG   0               ; RTEMP2 = unused
                          00000000    000D  3377          .LONG   0               ; RTEMP3 = unused
                          00000000    0011  3378          .LONG   0               ; RTEMP4 = unused
                          00000000    0015  3379          .LONG   0               ; RTEMP5 = INSTRUCTION ADDRESS
                                      0019  3380
                                      0019  3381 END_CPU_DATA
                                      00F9  3382
                                      00F9  3383 END_TEST_DATA
                                      00F9
                                      00F9       ;-------------------------------------------------------------------------
```

```
                00F9  3384 ENDTEST
```

```
0034          .SBTTL  "TEST   0D8     CM First IRDX Instruction Address Decode Test 1
0034  3386 ;*****************************************************************************
0034  3387 ;++
0034  3388 ;
0034  3389 ; FUNCTIONAL DESCRIPTION
0034  3390 ;
0034  3391 ;       This test makes sure the proper control store addresses are generated
0034  3392 ;       for the first compatiblity mode IRDX. The test is implemented by using
0034  3393 ;       two data tables. The first table consists of assembled VAX compatibil-
0034  3394 ;       ity mode macro instructions. The second table consists of expected
0034  3395 ;       control store addresses for the first IRDX.
0034  3396 ;
0034  3397 ; TEST ALGORITHM
0034  3398 ;
0034  3399 ;       This test is executed in the following manner:
0034  3400 ;
0034  3401 ;        1. Initialize the CPU.
0034  3402 ;        2. Execute micro-code to place the CPU into compatibility mode,
0034  3403 ;           disable the console CTRL-P interrupt and zeroes the MDR in order
0034  3404 ;           to allow the IRDX to function.
0034  3405 ;        3. Initialize the WD register with zero.
0034  3406 ;        4. Prepare a loop for 28 instructions.
0034  3407 ;        5. Load the WD register with a model opcode from the opcode table
0034  3408 ;           indexed by the loop count.
0034  3409 ;        6. Execute the micro-code to:
0034  3410 ;                1. Load an RTEMP with the data from the WD register.
0034  3411 ;                2. Load the VA with zero.
0034  3412 ;                3. Write the model instruction from RTEMP to cache.
0034  3413 ;                4. Increment the VA for the next address.
0034  3414 ;                5. Write zeros into this cache location to prevent cache
0034  3415 ;                   parity errors from trapping the IRD1 decode.
0034  3416 ;                6. Zero the PC which will initiate instruction prefetch.
0034  3417 ;                7. Execute a BUT/IRD1TST to load the IR and OSR.
0034  3418 ;                8. Execute a BUT/IRDX.
0034  3419 ;                9. Halt.
0034  3420 ;        7. Execute a masked comparison of the control store address trace
0034  3421 ;           register with the expected CS address.
0034  3422 ;        8. If the two are different signal an error.
0034  3423 ;        9. Otherwise continue looping from step 4 to step 8 till finished.
0034  3424 ;       10. Execute the micro-code to place the CPU back into native mode
0034  3425 ;           and enable the console CTRL-P interrupt.
0034  3426 ;       11. Exit.
0034  3427 ;
0034  3428 ; DATA PATTERNS
0034  3429 ;
0034  3430 ;       The input data patterns were derived from chapter 4.4 of the CPU
0034  3431 ;       micro-sequencer specification dated on 9/25/78. Because operand
0034  3432 ;       specifiers are embedded into compatibility mode instructions, certain
0034  3433 ;       instructions have more than one opcode decode. Model opcodes were
0034  3434 ;       generated by first using Chart 4.4.1B to obtain the base opcodes for
0034  3435 ;       each compatibility mode instruction. These opcodes were then enhanced
0034  3436 ;       to include all the possible decodes. The enhancements were obtained
0034  3437 ;       by changing the operands to specify both register and non-register
0034  3438 ;       modes and by using Chart 4.4.1A to modify the operands to get all
0034  3439 ;       possible instruction decodes.
0034  3440 ;
0034  3441 ;       The expected control store addresses where obtained from the
```

```
0034  3442 ;       Compatibility Mode Ird Rom Definition in the CPU micro-code.
0034  3443 ;
0034  3444 ;       The following is a table that shows the instruction being decoded for
0034  3445 ;       a range of loop counts:
0034  3446 ;
0034  3447 ;       NOTE! - This table is NOT the same as that for CM IRD1.
0034  3448 ;
0034  3449 ;       Loop            Instruction             Loop            Instruction
0034  3450 ;
0034  3451 ;       01 - 10         JSR                     96              FSUB
0034  3452 ;       11 - 12         CLR                     97              FMUL
0034  3453 ;       13 - 14         COM                     98              FDIV
0034  3454 ;       15 - 16         INC                     99 - A0         (unused)
0034  3455 ;       17 - 18         DEC                     A1 - A2         SOB
0034  3456 ;       19 - 1A         NEG                     A3 - BE         (reserved)
0034  3457 ;       1B - 1C         ADC                     BF              BR
0034  3458 ;       1D - 1E         SBC                     C0              BNE
0034  3459 ;       1F - 20         TST                     C1              BEQ
0034  3460 ;       21 - 22         ROR                     C2              BGE
0034  3461 ;       23 - 24         ROL                     C3              BLT
0034  3462 ;       25 - 26         ASR                     C4              BGT
0034  3463 ;       27 - 28         ASL                     C5              BLE
0034  3464 ;       29 - 2A         MARK                    C6              BPL
0034  3465 ;       2B - 2C         MFPI                    C7              BMI
0034  3466 ;       2D - 2E         MTPI                    C8              BHI
0034  3467 ;       2F - 30         SXT                     C9              BLOS
0034  3468 ;       31 - 40         (unused)                CA              BVC
0034  3469 ;       41 - 48         EMT                     CB              BVS
0034  3470 ;       49 - 50         TRAP                    CC              BCC, BHIS
0034  3471 ;       51 - 52         CLRB                    CD              BCS, BLO
0034  3472 ;       53 - 54         COMB                    CE              HALT
0034  3473 ;       55 - 56         INCB                    CF              WAIT
0034  3474 ;       57 - 58         DECB                    D0              RTI
0034  3475 ;       59 - 5A         NEGB                    D1              BPT
0034  3476 ;       5B - 5C         ADCB                    D2              IOT
0034  3477 ;       5D - 5E         SBCB                    D3              RESET
0034  3478 ;       5F - 60         TSTB                    D4              RTT
0034  3479 ;       61 - 62         RORB                    D5              (unused)
0034  3480 ;       63 - 64         ROLB                    D6 - D9         MOV
0034  3481 ;       65 - 66         ASRB                    DA - DD         CMP
0034  3482 ;       67 - 68         ASLB                    DE - E1         BIT
0034  3483 ;       69 - 6A         (unused)                E2 - E5         BIC
0034  3484 ;       6B - 6C         MFPD                    E6 - E9         BIS
0034  3485 ;       6D - 6E         MTPD                    EA - ED         ADD
0034  3486 ;       6F - 80         (unused)                EE - F1         MOVB
0034  3487 ;       81 - 84         MUL                     F2 - F5         CMPB
0034  3488 ;       85 - 88         DIV                     F6 - F9         BITB
0034  3489 ;       89 - 8C         ASH                     FA - FD         BICB
0034  3490 ;       8D - 90         ASHC                    FE - 100        BISB
0034  3491 ;       91 - 94         XOR
0034  3492 ;       95              FADD
0034  3493 ;
0034  3494 ; LOGICAL DESCRIPTION
0034  3495 ;
0034  3496 ;       During an IRD1 in compatibility mode the CCS ROMs are fed addresses
0034  3497 ;       from the Compatibility Mode IRD1 and IRDX ROMs on the DPM. The DPM ROMs
0034  3498 ;       are addressed according to the following scheme:
```

```
0034  3499 ;
0034  3500 ;       Source          ROM Address Bits        Comment
0034  3501 ;
0034  3502 ;       CIR <7:0>       <10:3>                  Encoded opcode generated by the
0034  3503 ;                                               IRD gate array.
0034  3504 ;
0034  3505 ;       IRD1            <2>                     Distinguishes between the 1st
0034  3506 ;                                               operand and the 2nd operand or
0034  3507 ;                                               execution flows. 1 during IRD1
0034  3508 ;                                               0 otherwise.
0034  3509 ;
0034  3510 ;       IRD Counter <0> <1>                     Distinguishes between the 2nd
0034  3511 ;                                               operand and execution flows.
0034  3512 ;                                               0 when decoding OS2, 1 when
0034  3513 ;                                               transferring to execution flows
0034  3514 ;
0034  3515 ;       Register Mode   <0>                     1 if the operand being eval-
0034  3516 ;                                               uated specifies register mode.
0034  3517 ;                                               Generated by the IRD and PHB
0034  3518 ;                                               gate arrays.
0034  3519 ;
0034  3520 ;       For zero and certain single operand instructions the addresses gener-
0034  3521 ;       ated will be execution flows in the micro-code. For the rest, operand
0034  3522 ;       specifier evaulation addresses will be generated.
0034  3523 ;
0034  3524 ; ASSUMPTIONS
0034  3525 ;
0034  3526 ;       This test assumes the following logic has already been tested; the
0034  3527 ;       cache, XB's, the IR and the OSR.
0034  3528 ;
0034  3529 ; ERROR DESCRIPTION
0034  3530 ;
0034  3531 ;       Below is a table of possible error symptoms and their most likely
0034  3532 ;       causes:
0034  3533 ;
0034  3534 ;               Symptom                                 Probable Cause
0034  3535 ;       +-------------------------------+-------------------------------+
0034  3536 ;       | The received control store    |                               |
0034  3537 ;       | address is bad for one or sev-| One or several bad CM IRD1 and|
0034  3538 ;       | eral instructions, or shows   | IRDX ROMs.                    |
0034  3539 ;       | certain bits stuck at 1 or 0. |                               |
0034  3540 ;       +-------------------------------+-------------------------------+
0034  3541 ;       | The received control store    |                               |
0034  3542 ;       | address is bad for every      |                               |
0034  3543 ;       | instruction or branches to the| A bad IRD gate array.         |
0034  3544 ;       | CM.BAD.IRD address in the     |                               |
0034  3545 ;       | micro-code.                   |                               |
0034  3546 ;       +-------------------------------+-------------------------------+
0034  3547 ;       | Register mode fails to assert |                               |
0034  3548 ;       | causing the wrong address to  | A bad IRD or PHB gate array.  |
0034  3549 ;       | be generated.                 |                               |
0034  3550 ;       +-------------------------------+-------------------------------+
0034  3551 ;
0034  3552 ;       On error this test will call out the IRD and PHB gate arrays and will
0034  3553 ;       also type out the following data:
0034  3554 ;
0034  3555 ;       EXPECTED CONTROL STORE ADDRESS
```

```
                        0034  3556 ;       RECEIVED CONTROL STORE ADDRESS
                        0034  3557 ;       LOOP COUNT
                        0034  3558 ;       DATA RESERVED FOR INSTRUCTION
                        0034  3559 ;       OPCODE OF THE INSTRUCTION BEING DECODED
                        0034  3560 ;
                        0034  3561 ;--
                        0034  3562 ;*******************************************************************************
                        0034  3563 ;///////////////////////////////////////////////////////////////////////////////
                        0034  3564
                        0034  3565         INITIALIZE
                        0036  3566         FETCH           0                       ; CPU INTO COMP. MODE, MDR_0
                        003B  3567         BRSTCLK         5                       ; AND DISABLE ^P INT FOR IRDX
                        003F  3568         LOADREG         WDREG,1$,,L             ; INITIALIZE THE WD REG.
                        0047  3569         EXPUTRAP                                ; EXPECT NON-DCS ADDRESSES
                        0049  3570         LOOP            I,1,256                 ; LOOP FOR 256 OPCODES
                        0050  3571         LOADREG         WDREG,2$,I,W            ; LOAD A MODEL OPCODE
                        0058  3572         ERRLOOP                                 ; LOOP ON ERROR FROM HERE
                        005A  3573         FETCH           6                       ; EXECUTE THE IRDX
                        005F  3574         BRSTCLK         <^X12>,INH              ; ...
                        0063  3575         CMPREGMSK       CSTRP,3$,I,4$,,W        ; IS THE CS ADDRESS CORRECT?
                        006F  3576         IFERROR         2,<<IRD><PHB>>          ; IF NOT SIGNAL AN ERROR
                        0076  3577         ENDLOOP         I                       ; CONTINUE LOOP UNTIL FINISHED
                        0079  3578         FETCH           <^X13>                  ; PLACE CPU BACK TO NATIVE MODE
                        007E  3579         BRSTCLK         <^X18>                  ; AND ENABLE CONSOLE INT.
                        0082  3580         SKIP                                    ; END OF TEST
                        0089  3581
                        0089  3582 BEGIN_TEST_DATA
                        0089
                        0089         ;-------------------------------------------------------------------------
                        0089  3583
                   00000000  0089  3584 1$:      .LONG   0       ; LONGWORD TO INITIALIZE THE WD REGISTER
                        008D  3585 ;
                        008D  3586 ; Table of test opcodes.
                        008D  3587 ;
                        008D  3588 ;       The numbers to the right of the semicolons indicates the range
                        008D  3589 ;       of the loop count for those patterns.
                        008D  3590 ;
08C0 0888 0880 0848 0840 0808 0800  008D  3591 2$:      .WORD   ^X0800,^X0808,^X0840,^X0848,^X0880,^X0888,^X08C0 ; 01 - 07
0988 0980 0948 0940 0908 0900 08C8  009B  3592          .WORD   ^X08C8,^X0900,^X0908,^X0940,^X0948,^X0980,^X0988 ; 08 - 0E
0A80 0A48 0A40 0A08 0A00 09C8 09C0  00A9  3593          .WORD   ^X09C0,^X09C8,^X0A00,^X0A08,^X0A40,^X0A48,^X0A80 ; 0F - 15
0B48 0B40 0B08 0B00 0AC8 0AC0 0A88  00B7  3594          .WORD   ^X0A88,^X0AC0,^X0AC8,^X0B00,^X0B08,^X0B40,^X0B48 ; 16 - 1C
0C40 0C08 0C00 0BC8 0BC0 0B88 0B80  00C5  3595          .WORD   ^X0B80,^X0B88,^X0BC0,^X0BC8,^X0C00,^X0C08,^X0C40 ; 1D - 23
0D08 0D00 0CC8 0CC0 0C88 0C80 0C48  00D3  3596          .WORD   ^X0C48,^X0C80,^X0C88,^X0CC0,^X0CC8,^X0D00,^X0D08 ; 24 - 2A
0E00 0DC8 0DC0 0D88 0D80 0D48 0D40  00E1  3597          .WORD   ^X0D40,^X0D48,^X0D80,^X0D88,^X0DC0,^X0DC8,^X0E00 ; 2B - 31
0EC8 0EC0 0E88 0E80 0E48 0E40 0E08  00EF  3598          .WORD   ^X0E08,^X0E40,^X0E48,^X0E80,^X0E88,^X0EC0,^X0EC8 ; 32 - 38
0FC0 0F88 0F80 0F48 0F40 0F08 0F00  00FD  3599          .WORD   ^X0F00,^X0F08,^X0F40,^X0F48,^X0F80,^X0F88,^X0FC0 ; 39 - 3F
8888 8880 8848 8840 8808 8800 0FC8  010B  3600          .WORD   ^X0FC8,^X8800,^X8808,^X8840,^X8848,^X8880,^X8888 ; 40 - 46
8980 8948 8940 8908 8900 88C8 88C0  0119  3601          .WORD   ^X88C0,^X88C8,^X8900,^X8908,^X8940,^X8948,^X8980 ; 47 - 4D
8A48 8A40 8A08 8A00 89C8 89C0 8988  0127  3602          .WORD   ^X8988,^X89C0,^X89C8,^X8A00,^X8A08,^X8A40,^X8A48 ; 4E - 54
8B40 8B08 8B00 8AC8 8AC0 8A88 8A80  0135  3603          .WORD   ^X8A80,^X8A88,^X8AC0,^X8AC8,^X8B00,^X8B08,^X8B40 ; 55 - 5B
8C08 8C00 8BC8 8BC0 8B88 8B80 8B48  0143  3604          .WORD   ^X8B48,^X8B80,^X8B88,^X8BC0,^X8BC8,^X8C00,^X8C08 ; 5C - 62
8D00 8CC8 8CC0 8C88 8C80 8C48 8C40  0151  3605          .WORD   ^X8C40,^X8C48,^X8C80,^X8C88,^X8CC0,^X8CC8,^X8D00 ; 63 - 69
8DC8 8DC0 8D88 8D80 8D48 8D40 8D08  015F  3606          .WORD   ^X8D08,^X8D40,^X8D48,^X8D80,^X8D88,^X8DC0,^X8DC8 ; 6A - 70
8EC0 8E88 8E80 8E48 8E40 8E08 8E00  016D  3607          .WORD   ^X8E00,^X8E08,^X8E40,^X8E48,^X8E80,^X8E88,^X8EC0 ; 71 - 77
8F88 8F80 8F48 8F40 8F08 8F00 8EC8  017B  3608          .WORD   ^X8EC8,^X8F00,^X8F08,^X8F40,^X8F48,^X8F80,^X8F88 ; 78 - 7E
7200 7088 7080 7008 7000 8FC8 8FC0  0189  3609          .WORD   ^X8FC0,^X8FC8,^X7000,^X7008,^X7080,^X7088,^X7200 ; 7F - 85
7488 7480 7408 7400 7288 7280 7208  0197  3610          .WORD   ^X7208,^X7280,^X7288,^X7400,^X7408,^X7480,^X7488 ; 86 - 8C
```

```
 7880 7808 7800 7688 7680 7608 7600  01A5  3611          .WORD   ^X7600,^X7608,^X7680,^X7688,^X7800,^X7808,^X7880 ; 8D - 93
 7A28 7A20 7A18 7A10 7A08 7A00 7888  01B3  3612          .WORD   ^X7888,^X7A00,^X7A08,^X7A10,^X7A18,^X7A20,^X7A28 ; 94 - 9A
 7E00 7CA8 7CA0 7C28 7C20 7AA8 7AA0  01C1  3613          .WORD   ^X7AA0,^X7AA8,^X7C20,^X7C28,^X7CA0,^X7CA8,^X7E00 ; 9B - A1
 F208 F200 F088 F080 F008 F000 7E80  01CF  3614          .WORD   ^X7E80,^XF000,^XF008,^XF080,^XF088,^XF200,^XF208 ; A2 - A8
 F680 F608 F600 F480 F400 F288 F280  01DD  3615          .WORD   ^XF280,^XF288,^XF400,^XF480,^XF600,^XF608,^XF680 ; A9 - AF
 FA08 FA00 F888 F880 F808 F800 F688  01EB  3616          .WORD   ^XF688,^XF800,^XF808,^XF880,^XF888,^XFA00,^XFA08 ; B0 - B6
 FE00 FC88 FC80 FC08 FC00 FA88 FA80  01F9  3617          .WORD   ^XFA80,^XFA88,^XFC00,^XFC08,^XFC80,^XFC88,^XFE00 ; B7 - BD
 0600 0500 0400 0300 0200 0100 FE80  0207  3618          .WORD   ^XFE80,^X0100,^X0200,^X0300,^X0400,^X0500,^X0600 ; BE - C4
 8500 8400 8300 8200 8100 8000 0700  0215  3619          .WORD   ^X0700,^X8000,^X8100,^X8200,^X8300,^X8400,^X8500 ; C5 - CB
 0004 0003 0002 0001 0000 8700 8600  0223  3620          .WORD   ^X8600,^X8700,^X0000,^X0001,^X0002,^X0003,^X0004 ; CC - D2
 1088 1080 1008 1000 0007 0006 0005  0231  3621          .WORD   ^X0005,^X0006,^X0007,^X1000,^X1008,^X1080,^X1088 ; D3 - D9
 3080 3008 3000 2088 2080 2008 2000  023F  3622          .WORD   ^X2000,^X2008,^X2080,^X2088,^X3000,^X3008,^X3080 ; DA - E0
 5008 5000 4088 4080 4008 4000 3088  024D  3623          .WORD   ^X3088,^X4000,^X4008,^X4080,^X4088,^X5000,^X5008 ; E1 - E7
 9000 6088 6080 6008 6000 5088 5080  025B  3624          .WORD   ^X5080,^X5088,^X6000,^X6008,^X6080,^X6088,^X9000 ; E8 - EE
 A088 A080 A008 A000 9088 9080 9008  0269  3625          .WORD   ^X9008,^X9080,^X9088,^XA000,^XA008,^XA080,^XA088 ; EF - F5
 C080 C008 C000 B088 B080 B008 B000  0277  3626          .WORD   ^XB000,^XB008,^XB080,^XB088,^XC000,^XC008,^XC080 ; F6 - FC
           D080 D008 D000 C088       0285  3627          .WORD   ^XC088,^XD000,^XD008,^XD080                       ; FD - 100
                                     028D  3628 ;
                                     028D  3629 ; Table of control store addresses.
                                     028D  3630 ;
                                     028D  3631 ;       The numbers to the right of the semicolons indicates the range
                                     028D  3632 ;       of the loop count for those patterns.
                                     028D  3633 ;
 03E6 03E6 03E6 03E6 03E6 03E6 03E6  028D  3634 3$:      .WORD   ^X03E6,^X03E6,^X03E6,^X03E6,^X03E6,^X03E6,^X03E6 ; 01 - 07
 03E6 03E6 03E6 03E6 03E6 03E6 03E6  029B  3635          .WORD   ^X03E6,^X03E6,^X03E6,^X03E6,^X03E6,^X03E6,^X03E6 ; 08 - 0E
 03F8 03CF 03F8 03CA 03F8 03E6 03E6  02A9  3636          .WORD   ^X03E6,^X03E6,^X03F8,^X03CA,^X03F8,^X03CF,^X03F8 ; 0F - 15
 03D5 03F8 03CD 03F8 03CB 03F8 03CC  02B7  3637          .WORD   ^X03CC,^X03F8,^X03CB,^X03F8,^X03CD,^X03F8,^X03D5 ; 16 - 1C
 03F8 03D9 03F8 03CE 03F8 03D6 03F8  02C5  3638          .WORD   ^X03F8,^X03D6,^X03F8,^X03CE,^X03F8,^X03D9,^X03F8 ; 1D - 23
 03F8 03F8 03D1 03F8 03D0 03F8 03D8  02D3  3639          .WORD   ^X03D8,^X03F8,^X03D0,^X03F8,^X03D1,^X03F8,^X03F8 ; 24 - 2A
 03F8 03D7 03F8 03E9 03F8 03E8 03F8  02E1  3640          .WORD   ^X03F8,^X03E8,^X03F8,^X03E9,^X03F8,^X03D7,^X03F8 ; 2B - 31
 03F8 03F8 03F8 03F8 03F8 03F8 03F8  02EF  3641          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 32 - 38
 03F8 03F8 03F8 03F8 03F8 03F8 03F8  02FD  3642          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 39 - 3F
 03F8 03F8 03F8 03F8 03F8 03F8 03F8  030B  3643          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 40 - 46
 03F8 03F8 03F8 03F8 03F8 03F8 03F8  0319  3644          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 47 - 4D
 03CF 03F8 03CA 03F8 03F8 03F8 03F8  0327  3645          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03CA,^X03F8,^X03CF ; 4E - 54
 03F8 03CD 03F8 03CB 03F8 03CC 03F8  0335  3646          .WORD   ^X03F8,^X03CC,^X03F8,^X03CB,^X03F8,^X03CD,^X03F8 ; 55 - 5B
 03DA 03F8 03CE 03F8 03D6 03F8 03D5  0343  3647          .WORD   ^X03D5,^X03F8,^X03D6,^X03F8,^X03CE,^X03F8,^X03DA ; 5C - 62
 03F8 03D1 03F8 03D0 03F8 03D8 03F8  0351  3648          .WORD   ^X03F8,^X03D8,^X03F8,^X03D0,^X03F8,^X03D1,^X03F8 ; 63 - 69
 03F8 03F8 03E9 03F8 03E8 03F8 03F8  035F  3649          .WORD   ^X03F8,^X03F8,^X03E8,^X03F8,^X03E9,^X03F8,^X03F8 ; 6A - 70
 03F8 03F8 03F8 03F8 03F8 03F8 03F8  036D  3650          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 71 - 77
 03F8 03F8 03F8 03F8 03F8 03F8 03F8  037B  3651          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 78 - 7E
 03E1 03E0 03E0 03E0 03E0 03F8 03F8  0389  3652          .WORD   ^X03F8,^X03F8,^X03E0,^X03E0,^X03E0,^X03E0,^X03E1 ; 7F - 85
 03D3 03D3 03D3 03D3 03E1 03E1 03E1  0397  3653          .WORD   ^X03E1,^X03E1,^X03E1,^X03D3,^X03D3,^X03D3,^X03D3 ; 86 - 8C
 02E0 02E2 02E0 03D4 03D4 03D4 03D4  03A5  3654          .WORD   ^X03D4,^X03D4,^X03D4,^X03D4,^X02E0,^X02E2,^X02E0 ; 8D - 93
 03F8 03F8 03F8 03F8 03F8 03F8 02E2  03B3  3655          .WORD   ^X02E2,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 94 - 9A
 03F8 03F8 03F8 03F8 03F8 03F8 03F8  03C1  3656          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 9B - A1
 03F8 03F8 03F8 03F8 03F8 03F8 03F8  03CF  3657          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; A2 - A8
 03F8 03F8 03F8 03F8 03F8 03F8 03F8  03DD  3658          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; A9 - AF
 03F8 03F8 03FA 03F8 03FA 03F8 03F8  03EB  3659          .WORD   ^X03F8,^X03F8,^X03FA,^X03F8,^X03FA,^X03F8,^X03F8 ; B0 - B6
 03F8 03F8 03F8 03F8 03F8 03F8 03F8  03F9  3660          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; B7 - BD
 03F8 03F8 03F8 03F8 03F8 03F8 03F8  0407  3661          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; BE - C4
 03F8 03F8 03F8 03F8 03F8 03F8 03F8  0415  3662          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; C5 - CB
 03F8 03F8 03F8 03F8 03F8 03F8 03F8  0423  3663          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; CC - D2
 02F2 00C5 02F2 00C5 03F8 03F8 03F8  0431  3664          .WORD   ^X03F8,^X03F8,^X03F8,^X00C5,^X02F2,^X00C5,^X02F2 ; D3 - D9
 0352 02D2 0352 02D2 034C 02D2 034C  043F  3665          .WORD   ^X034C,^X02D2,^X034C,^X02D2,^X0352,^X02D2,^X0352 ; DA - E0
 02E2 034E 02E2 0354 02E2 0354 02D2  044D  3666          .WORD   ^X02D2,^X0354,^X02E2,^X0354,^X02E2,^X034E,^X02E2 ; E1 - E7
 033A 02E2 0342 02E2 0342 02E2 034E  045B  3667          .WORD   ^X034E,^X02E2,^X0342,^X02E2,^X0342,^X02E2,^X033A ; E8 - EE
```

```
 02D2 034C 02D2 034C 02F2 033A 02F2  0469  3668          .WORD   ^X02F2,^X033A,^X02F2,^X034C,^X02D2,^X034C,^X02D2 ; EF - F5
 0354 02E2 0354 02D2 0352 02D2 0352  0477  3669          .WORD   ^X0352,^X02D2,^X0352,^X02D2,^X0354,^X02E2,^X0354 ; F6 - FC
           034E 02E2 034E 02E2       0485  3670          .WORD   ^X02E2,^X034E,^X02E2,^X034E                      ; FD - 100
                                     048D  3671
                               3FFF  048D  3672 4$:      .WORD   ^X3FFF          ; MASK FOR ADDRESS COMPARISON
                                     048F  3673
                                     048F  3674 BEGIN_CPU_DATA
                                     0002  3675
                                     0002  3676 DCS_DATA
                                     0001  3677
U 00. 7700,C800,0364,0300,0470,1801  0001  3678 ;% 00    NOP                                    ; GET THE CPU UP TO SPEED
U 01. 7700,3800,3FFF,FFFF,8470,1802  000C  3682 ;% 01    LONLIT_[80000000]                      ; PLACES CPU INTO COMP. MODE
U 02. 7700,C800,5BE4,03D4,0070,1803  0017  3686 ;% 02    WB_R[LONLIT],CCPSL/PSL_WB.CCBR_ALUS    ; ...
U 03. 7700,0800,5BE4,0308,2470,1804  0022  3690 ;% 03    WCTRL/LOADCRAR,WB_R[TEMP2]             ; DISABLES CTRL-P INTERRUPT
U 04. 7700,4800,5BE4,030C,2C70,1805  002D  3694 ;% 04    WCTRL/CONWRITE,WB_R[TEMP3]             ; ...
U 05. 7700,C800,0364,0300,4E70,1806  0038  3698 ;% 05    MDR_0                                  ; ZERO THE MDR FOR THE IRDX
U 06. 7300,C800,0364,0300,0470,1807  0043  3702 ;% 06    STOP.CLOCK
                                     004E  3706
U 07. 7700,C800,0364,0300,0470,1808  004E  3707 ;% 07    NOP                                    ; GET THE CPU UP TO SPEED
U 08. 5700,4840,0364,0304,0470,1809  0059  3711 ;% 08    R[TEMP1]_RDM                           ; PUT MODEL OPCODE INTO RTEMP1
U 09. 7700,4800,5BE4,0314,4A70,180A  0064  3715 ;% 09    VA_R[TEMP5]                            ; ZERO THE VA
U 0A. 7701,C800,5BE4,0204,5590,180B  006F  3719 ;% 0A    WRITE.LONG R[TEMP1]                    ; WRITE OPCODE TO CACHE FOR XB0
U 0B. 7701,4800,5BE4,0204,55F0,180C  007A  3723 ;% 0B    WDR_UNROT(R[TEMP1]),BUS/PRB.RD,SIZE[LONG]
                                     0085  3727                                             ; HOLD FOR SECOND CYCLE
U 0C. 7700,4800,0364,0300,4470,180D  0085  3728 ;% 0C    VA_VA+4                                ; INCREMENT THE VA
U 0D. 7701,8800,5BE4,0214,5590,180E  0090  3732 ;% 0D    WRITE.LONG R[TEMP5]                    ; WRITE ZERO TO VALIDATE CACHE
U 0E. 7701,0800,5BE4,0214,55F0,180F  009B  3736 ;% 0E    WDR_UNROT(R[TEMP5]),BUS/PRB.RD,SIZE[LONG]
                                     00A6  3740                  ; HOLD FOR SECOND CYCLE
U 0F. 7700,4800,5BE4,0314,4870,1810  00A6  3741 ;% 0F    PC_R[TEMP5]                            ; INITIATE INSTRUCTION PREFETCH
U 10. 7700,4800,0364,1700,0470,1811  00B1  3745 ;% 10    BUT/IRD1TST                            ; LOAD THE IR AND OSR
U 11. 7700,0800,0364,0700,0470,1812  00BC  3749 ;% 11    BUT/IRDX                               ; EXECUTE AN IRDX
U 12. 7300,C800,0364,0300,0470,1813  00C7  3753 ;% 12    STOP.CLOCK                             ; HALT
                                     00D2  3757
U 13. 7700,4800,0364,0300,0470,1814  00D2  3758 ;% 13    NOP                                    ; GET THE CPU UP TO SPEED
U 14. 7700,0800,5BE4,0308,2470,1815  00DD  3762 ;% 14    WCTRL/LOADCRAR,WB_R[TEMP2]             ; ENABLE CTRL-P INTERRUPTS
U 15. 7700,8800,5BE4,0310,2C70,1816  00E8  3766 ;% 15    WCTRL/CONWRITE,WB_R[TEMP4]             ; ...
U 16. 7700,F800,7FFF,FFFF,8470,1817  00F3  3770 ;% 16    LONLIT_[0]                             ; PUT CPU BACK INTO NATIVE MODE
U 17. 7700,C800,5BE4,03D4,0070,1818  00FE  3774 ;% 17    WB_R[LONLIT],CCPSL/PSL_WB.CCBR_ALUS    ; ...
U 18. 7300,C800,0364,0300,0470,1819  0109  3778 ;% 18    STOP.CLOCK                             ; HALT
                                     0114  3782
                                     0114  3783 RTEMP_DATA
                                     0001  3784
                           00000000  0001  3785          .LONG   0               ; RTEMP0 = reserved for instruction use
                           00000000  0005  3786          .LONG   0               ; RTEMP1 = INSTRUCTION OPCODE
                           00C00000  0009  3787          .LONG   ^X00C00000      ; RTEMP2 = CRAR COMMAND AND STATUS
                           00400000  000D  3788          .LONG   ^X00400000      ; RTEMP3 = DISABLE INTERRUPT
                           00900000  0011  3789          .LONG   ^X00900000      ; RTEMP4 = ENABLE INTERRUPT
                           00000000  0015  3790          .LONG   0               ; RTEMP5 = ADDRESS OF THE MODEL INSTRUCTION
                                     0019  3791
                                     0019  3792 END_CPU_DATA
                                     048F  3793
                                     048F  3794 END_TEST_DATA
                                     048F
                                     048F        ;-------------------------------------------------------------------------
                                     048F  3795 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
0034          .SBTTL  "TEST   0D9     CM First IRDX Instruction Address Decode Test 2
0034  3797 ;*****************************************************************************
0034  3798 ;++
0034  3799 ;
0034  3800 ; FUNCTIONAL DESCRIPTION
0034  3801 ;
0034  3802 ;       This test makes sure the proper control store addresses are generated
0034  3803 ;       for the first compatiblity mode IRDX. The test is implemented by using
0034  3804 ;       two data tables. The first table consists of assembled VAX compatibil-
0034  3805 ;       ity mode macro instructions. The second table consists of expected
0034  3806 ;       control store addresses for the first IRDX. This test decodes those
0034  3807 ;       instructions not done in test one due to insufficient RDM memory space.
0034  3808 ;
0034  3809 ; TEST ALGORITHM
0034  3810 ;
0034  3811 ;       This test is executed in the following manner:
0034  3812 ;
0034  3813 ;        1. Initialize the CPU.
0034  3814 ;        2. Execute micro-code to place the CPU into compatibility mode,
0034  3815 ;           disable the console CTRL-P interrupt and zeroes the MDR in order
0034  3816 ;           to allow the IRDX to function.
0034  3817 ;        3. Initialize the WD register with zero.
0034  3818 ;        4. Prepare a loop for 30 instructions.
0034  3819 ;        5. Load the WD register with a model opcode from the opcode table
0034  3820 ;           indexed by the loop count.
0034  3821 ;        6. Execute the micro-code to:
0034  3822 ;                1. Load RTEMP1 with the data from the WD register.
0034  3823 ;                2. Load the VA with zero.
0034  3824 ;                3. Write the model instruction from RTEMP1 to cache.
0034  3825 ;                4. Increment the VA for the next address.
0034  3826 ;                5. Write zeros into this cache location to prevent cache
0034  3827 ;                   parity errors from trapping the IRD1 decode.
0034  3828 ;                6. Zero the PC which will initiate instruction prefetch.
0034  3829 ;                7. Execute a BUT/IRD1TST to load the IR and OSR.
0034  3830 ;                8. Execute a BUT/IRDX.
0034  3831 ;                9. Halt.
0034  3832 ;        7. Execute a masked comparison of the control store address trace
0034  3833 ;           register with the expected CS address.
0034  3834 ;        8. If the two are different signal an error.
0034  3835 ;        9. Otherwise continue looping from step 4 to step 8 till finished.
0034  3836 ;       10. Execute the micro-code to place the CPU back into native mode
0034  3837 ;           and enable the console CTRL-P interrupt.
0034  3838 ;       11. Exit.
0034  3839 ;
0034  3840 ; DATA PATTERNS
0034  3841 ;
0034  3842 ;       The input data patterns were derived from chapter 4.4 of the CPU
0034  3843 ;       micro-sequencer specification dated on 9/25/78. Because operand
0034  3844 ;       specifiers are embedded into compatibility mode instructions, certain
0034  3845 ;       instructions have more than one opcode decode. Model opcodes were
0034  3846 ;       generated by first using Chart 4.4.1B to obtain the base opcodes for
0034  3847 ;       each compatibility mode instruction. These opcodes were then enhanced
0034  3848 ;       to include all the possible decodes. The enhancements were obtained
0034  3849 ;       by changing the operands to specify both register and non-register
0034  3850 ;       modes and by using Chart 4.4.1A to modify the operands to get all
0034  3851 ;       possible instruction decodes.
0034  3852 ;
```

```
0034  3853 ;       The expected control store addresses where obtained from the
0034  3854 ;       Compatibility Mode Ird Rom Definition in the CPU micro-code.
0034  3855 ;
0034  3856 ;       The following is a table that shows the instruction being decoded for
0034  3857 ;       a range of loop counts:
0034  3858 ;
0034  3859 ;       NOTE! - This table is NOT the same as that for CM IRD1.
0034  3860 ;
0034  3861 ;       Loop            Instruction
0034  3862 ;
0034  3863 ;       01              BISB
0034  3864 ;       02 - 05         SUB
0034  3865 ;       06 - 0D         JMP
0034  3866 ;       0E              RTS
0034  3867 ;       0F - 10         (unused)
0034  3868 ;       11              SPL
0034  3869 ;       12              NOP
0034  3870 ;       13 - 16         (reserved)
0034  3871 ;       17 - 1E         SWAB
0034  3872 ;
0034  3873 ; LOGICAL DESCRIPTION
0034  3874 ;
0034  3875 ;       During an IRD1 in compatibility mode the CCS ROMs are fed addresses
0034  3876 ;       from the Compatibility Mode IRD1 and IRDX ROMs on the DPM. The DPM ROMs
0034  3877 ;       are addressed according to the following scheme:
0034  3878 ;
0034  3879 ;       Source          ROM Address Bits        Comment
0034  3880 ;
0034  3881 ;       CIR <7:0>       <10:3>                  Encoded opcode generated by the
0034  3882 ;                                               IRD gate array.
0034  3883 ;
0034  3884 ;       IRD1            <2>                     Distinguishes between the 1st
0034  3885 ;                                               operand and the 2nd operand or
0034  3886 ;                                               execution flows. 1 during IRD1
0034  3887 ;                                               0 otherwise.
0034  3888 ;
0034  3889 ;       IRD Counter <0> <1>                     Distinguishes between the 2nd
0034  3890 ;                                               operand and execution flows.
0034  3891 ;                                               0 when decoding OS2, 1 when
0034  3892 ;                                               transferring to execution flows
0034  3893 ;
0034  3894 ;       Register Mode   <0>                     1 if the operand being eval-
0034  3895 ;                                               uated specifies register mode.
0034  3896 ;                                               Generated by the IRD and PHB
0034  3897 ;                                               gate arrays.
0034  3898 ;
0034  3899 ;       For zero and certain single operand instructions the addresses gener-
0034  3900 ;       ated will be execution flows in the micro-code. For the rest, operand
0034  3901 ;       specifier evaulation addresses will be generated.
0034  3902 ;
0034  3903 ; ASSUMPTIONS
0034  3904 ;
0034  3905 ;       This test assumes the following logic has already been tested; the
0034  3906 ;       cache, XB's, the IR and the OSR.
0034  3907 ;
0034  3908 ; ERROR DESCRIPTION
0034  3909 ;
```

```
                 0034  ;       Below is a table of possible error symptoms and their most likely
                 0034  ;       causes:
                 0034  ;
                 0034  ;               Symptom                          Probable Cause
                 0034  ;       +-------------------------------+-------------------------------+
                 0034  ;       | The received control store    |                               |
                 0034  ;       | address is bad for one or sev-| One or several bad CM IRD1 and|
                 0034  ;       | eral instructions, or shows   | IRDX ROMs.                    |
                 0034  ;       | certain bits stuck at 1 or 0. |                               |
                 0034  ;       +-------------------------------+-------------------------------+
                 0034  ;       | The received control store    |                               |
                 0034  ;       | address is bad for every      |                               |
                 0034  ;       | instruction or branches to the| A bad IRD gate array.         |
                 0034  ;       | CM.BAD.IRD address in the     |                               |
                 0034  ;       | micro-code.                   |                               |
                 0034  ;       +-------------------------------+-------------------------------+
                 0034  ;       | Register mode fails to assert |                               |
                 0034  ;       | causing the wrong address to  | A bad IRD or PHB gate array.  |
                 0034  ;       | be generated.                 |                               |
                 0034  ;       +-------------------------------+-------------------------------+
                 0034  ;
                 0034  ;       On error this test will call out the IRD and PHB gate arrays and will
                 0034  ;       also type out the following data:
                 0034  ;
                 0034  ;       EXPECTED CONTROL STORE ADDRESS
                 0034  ;       RECEIVED CONTROL STORE ADDRESS
                 0034  ;       LOOP COUNT
                 0034  ;       DATA RESERVED FOR INSTRUCTION
                 0034  ;       OPCODE OF THE INSTRUCTION BEING DECODED
                 0034  ;
                 0034  ;--
                 0034  ;*******************************************************************************
                 0034  ;///////////////////////////////////////////////////////////////////////////////
                 0034
                 0034          INITIALIZE                                  ; INITIALIZE THE CPU
                 0036          FETCH           0                           ; CPU INTO COMP. MODE, MDR_0
                 003B          BRSTCLK         6                           ; AND DISABLE CONSOLE INTERRUPT
                 003F          LOADREG         WDREG,1$,,L                 ; INITIALIZE THE WD REG.
                 0047          EXPUTRAP                                    ; EXPECT NON-DCS ADDRESSES
                 0049          LOOP            I,1,30                      ; LOOP FOR 30 OPCODES
                 0050          LOADREG         WDREG,2$,I,W                ; LOAD A MODEL OPCODE
                 0058          ERRLOOP                                     ; LOOP ON ERROR FROM HERE
                 005A          FETCH           7                           ; EXECUTE THE IRDX
                 005F          BRSTCLK         <^X12>,INH                  ; ...
                 0063          CMPREGMSK       CSTRP,3$,I,4$,,W            ; IS THE CS ADDRESS CORRECT?
                 006F          IFERROR         2,<<IRD><PHB>>              ; IF NOT SIGNAL AN ERROR
                 0076          ENDLOOP         I                           ; CONTINUE LOOP UNTIL FINISHED
                 0079          FETCH           <^X13>                      ; PLACE CPU BACK INTO NATIVE
                 007E          BRSTCLK         <^X18>                      ; MODE AND ENABLE CONSOLE INT.
                 0082          SKIP                                        ; END OF TEST
                 0089
                 0089  BEGIN_TEST_DATA
                 0089
                 0089  ;-----------------------------------------------------------------------
                 0089
00000000         0089  1$:     .LONG   0       ; LONGWORD TO INITIALIZE THE WD REGISTER
                 008D  ;
```

```
                                    008D  3965 ; Table of test opcodes.
                                    008D  3966 ;
                                    008D  3967 ;       The numbers to the right of the semicolons indicates the range
                                    008D  3968 ;       of the loop count for those patterns.
                                    008D  3969 ;
 0048 0040 E088 E080 E008 E000 D088 008D  3970 2$:     .WORD   ^XD088,^XE000,^XE008,^XE080,^XE088,^X0040,^X0048 ; 01 - 07
 0080 0078 0070 0068 0060 0058 0050 009B  3971         .WORD   ^X0050,^X0058,^X0060,^X0068,^X0070,^X0078,^X0080 ; 08 - 0E
 00B0 00A9 00A1 00A0 0098 0090 0088 00A9  3972         .WORD   ^X0088,^X0090,^X0098,^X00A0,^X00A1,^X00A9,^X00B0 ; 0F - 15
 00E8 00E0 00D8 00D0 00C8 00C0 00B8 00B7  3973         .WORD   ^X00B8,^X00C0,^X00C8,^X00D0,^X00D8,^X00E0,^X00E8 ; 16 - 1C
                    00F8 00F0       00C5  3974         .WORD   ^X00F0,^X00F8                                    ; 1D - 1E
                                    00C9  3975 ;
                                    00C9  3976 ; Table of control store addresses.
                                    00C9  3977 ;
                                    00C9  3978 ;       The numbers to the right of the semicolons indicates the range
                                    00C9  3979 ;       of the loop count for those patterns.
                                    00C9  3980 ;
 03E7 03F8 02E2 034A 02E2 034A 02E2 00C9  3981 3$:     .WORD   ^X02E2,^X034A,^X02E2,^X034A,^X02E2,^X03F8,^X03E7 ; 01 - 07
 03F8 03E7 03E7 03E7 03E7 03E7 03E7 00D7  3982         .WORD   ^X03E7,^X03E7,^X03E7,^X03E7,^X03E7,^X03E7,^X03F8 ; 08 - 0E
 03F8 03F8 03F8 03F8 03F8 03F8 03F8 00E5  3983         .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 0F - 15
 03DB 03DB 03DB 03DB 03DB 03F8 03F8 00F3  3984         .WORD   ^X03F8,^X03F8,^X03DB,^X03DB,^X03DB,^X03DB,^X03DB ; 16 - 1C
                    03DB 03DB       0101  3985         .WORD   ^X03DB,^X03DB                                    ; 1D - 1E
                                    0105  3986
                         3FFF       0105  3987 4$:     .WORD   ^X3FFF  ; MASK FOR ADDRESS COMPARISON
                                    0107  3988
                                    0107  3989 BEGIN_CPU_DATA
                                    0002  3990
                                    0002  3991 DCS_DATA
                                    0001  3992
U 00, 7700,C800,0364,0300,0470,1801 0001  3993 ;% 00   NOP                                                    ; GET CPU UP TO SPEED
U 01, 7700,3800,3FFF,FFFF,8470,1802 000C  3997 ;% 01   LONLIT_[80000000]                                       ; PLACES CPU INTO COMP. MODE
U 02, 7700,C800,5BE4,03D4,0070,1803 0017  4001 ;% 02   WB_R[LONLIT],CCPSL/PSL_WB.CCBR_ALUS                    ; ...
U 03, 7700,0800,5BE4,0308,2470,1804 0022  4005 ;% 03   WCTRL/LOADCRAR,WB_R[TEMP2]                            ; DISABLE CONSOLE INTERRUPT
U 04, 7700,4800,5BE4,030C,2C70,1805 002D  4009 ;% 04   WCTRL/CONWRITE,WB_R[TEMP3]                            ; ...
U 05, 7700,C800,0364,0300,4E70,1806 0038  4013 ;% 05   MDR_0                                                 ; ZERO THE MDR
U 06, 7300,C800,0364,0300,0470,1807 0043  4017 ;% 06   STOP.CLOCK                                            ; HALT
                                    004E  4021
U 07, 7700,C800,0364,0300,0470,1808 004E  4022 ;% 07   NOP                                                    ; GET THE CPU UP TO SPEED
U 08, 5700,4840,0364,0304,0470,1809 0059  4026 ;% 08   R[TEMP1]_RDM                                          ; PUT MODEL OPCODE INTO RTEMP1
U 09, 7700,4800,5BE4,0314,4A70,180A 0064  4030 ;% 09   VA_R[TEMP5]                                          ; ZERO THE VA
U 0A, 7701,C800,5BE4,0204,5590,180B 006F  4034 ;% 0A   WRITE.LONG R[TEMP1]                                  ; WRITE OPCODE TO CACHE FOR XB0
U 0B, 7701,4800,5BE4,0204,55F0,180C 007A  4038 ;% 0B   WDR_UNROT(R[TEMP1]),BUS/PRB.RD,SIZE[LONG]
                                    0085  4042                                                             ; HOLD FOR SECOND CYCLE
U 0C, 7700,4800,0364,0300,4470,180D 0085  4043 ;% 0C   VA_VA+4                                              ; INCREMENT THE VA
U 0D, 7701,8800,5BE4,0214,5590,180E 0090  4047 ;% 0D   WRITE.LONG R[TEMP5]                                  ; VALIDATE THE CACHE FOR XB1
U 0E, 7701,0800,5BE4,0214,55F0,180F 009B  4051 ;% 0E   WDR_UNROT(R[TEMP5]),BUS/PRB.RD,SIZE[LONG]
                                    00A6  4055                                                             ; HOLD FOR SECOND CYCLE
U 0F, 7700,4800,5BE4,0314,4870,1810 00A6  4056 ;% 0F   PC_R[TEMP5]                                          ; INITIIATE INSTRUCION PREFETCH
U 10, 7700,4800,0364,1700,0470,1811 00B1  4060 ;% 10   BUT/IRD1TST                                         ; LOAD THE IR AND OSR
U 11, 7700,0800,0364,0700,0470,1812 00BC  4064 ;% 11   BUT/IRDX                                            ; EXECUTE AN IRDX
U 12, 7300,C800,0364,0300,0470,1813 00C7  4068 ;% 12   STOP.CLOCK                                          ; HALT
                                    00D2  4072
U 13, 7700,4800,0364,0300,0470,1814 00D2  4073 ;% 13   NOP                                                    ; GET THE CPU UP TO SPEED
U 14, 7700,0800,5BE4,0308,2470,1815 00DD  4077 ;% 14   WCTRL/LOADCRAR,WB_R[TEMP2]                            ; ENABLE CONSOLE INTERRUPTS
U 15, 7700,8800,5BE4,0310,2C70,1816 00E8  4081 ;% 15   WCTRL/CONWRITE,WB_R[TEMP4]                            ; ...
U 16, 7700,F800,7FFF,FFFF,8470,1817 00F3  4085 ;% 16   LONLIT_[0]                                           ; PUT CPU BACK INTO NATIVE MDOE
U 17, 7700,C800,5BE4,03D4,0070,1818 00FE  4089 ;% 17   WB_R[LONLIT],CCPSL/PSL_WB.CCBR_ALUS                    ; ...
U 18, 7300,C800,0364,0300,0470,1819 0109  4093 ;% 18   STOP.CLOCK                                          ; HALT
```

N 3


```
                    0114  4097
                    0114  4098 RTEMP_DATA
                    0001  4099
00000000            0001  4100          .LONG   0               ; RTEMP0 = reserved for instruction use
00000000            0005  4101          .LONG   0               ; RTEMP1 = INSTRUCTION OPCODE
00C00000            0009  4102          .LONG   ^X00C00000      ; RTEMP2 = CRAR RECV STATUS
00400000            000D  4103          .LONG   ^X00400000      ; RTEMP3 = DISABLES INTERRUPT
00900000            0011  4104          .LONG   ^X00900000      ; RTEMP4 = ENABLES INTERRUPT
00000000            0015  4105          .LONG   0               ; RTEMP5 = INSTRUCTION ADDRESS
                    0019  4106
                    0019  4107 END_CPU_DATA
                    0107  4108
                    0107  4109 END_TEST_DATA
                    0107
                    0107        ;-------------------------------------------------------------------------
                    0107  4110 ENDTEST
```

```
0035        .SBTTL  "TEST   0DA     CM Second IRDX Instruction Address Decode Test 1
0035  4112 ;*******************************************************************************
0035  4113 ;++
0035  4114 ;
0035  4115 ; FUNCTIONAL DESCRIPTION
0035  4116 ;
0035  4117 ;       This test makes sure the proper control store addresses are generated
0035  4118 ;       for the second compatiblity mode IRDX. The test is implemented by using
0035  4119 ;       two data tables. The first table consists of assembled VAX compatibil-
0035  4120 ;       ity mode macro instructions. The other table consists of expected
0035  4121 ;       control store addresses for the second IRDX.
0035  4122 ;
0035  4123 ; TEST ALGORITHM
0035  4124 ;
0035  4125 ;       The IRDX micro-order will assert addresses onto the CS address bus
0035  4126 ;       which will cause the RDM to stop the processor clock. Thus if the first
0035  4127 ;       IRDX is executed using the FETCH and BRSTCLK pseudo-ops, the clock will
0035  4128 ;       stop before the IRDX is finished. This causes bad addresses to be
0035  4129 ;       generated for some of the macro instructions being decoded. To prevent
0035  4130 ;       this from happening, the LOADREG pseudo-op is used to execute the
0035  4131 ;       micro-code that fetches the macro instructions, loads the IR and OSR
0035  4132 ;       and does the first IRDX. The second IRDX can be executed using the
0035  4133 ;       FETCH and BRSTCLK pseudo-ops since the address that is generated will
0035  4134 ;       not depend on whether that micro-order finishes execution.
0035  4135 ;
0035  4136 ;       This test is executed in the following manner:
0035  4137 ;
0035  4138 ;        1. Initialize the CPU.
0035  4139 ;        2. Execute micro-code to place the CPU into compatibility mode,
0035  4140 ;           disable the console CTRL-P interrupt and zero the MDR in order
0035  4141 ;           to allow the IRDX to function.
0035  4142 ;        3. Initialize the WD register with zero.
0035  4143 ;        4. Prepare a loop for 239 instructions.
0035  4144 ;        5. Load the WD register with a model opcode from the opcode table
0035  4145 ;           indexed by the loop count.
0035  4146 ;        6. Take the CPU off the CS address bus and hold the CPU clock
0035  4147 ;           off by setting the bits uADINH, CLKCN1, CLKCN0, PARSTP, and PARCHK
0035  4148 ;           in the DCS Control Register (DCSCR).
0035  4149 ;        7. Load the DCS Address Counter (DCSAD) with the starting location of
0035  4150 ;           the test micro-code.
0035  4151 ;        8. Load the Control Store Address Match Register (CSAMR) with the
0035  4152 ;           starting address of the test micro-code to enable DCS onto the
0035  4153 ;           CPU control store bus.
0035  4154 ;        9. Initialize the RDM Control Register (RDCTRL) with zero.
0035  4155 ;       10. Set the bits TSTP (trap stop) and MSTP (master halt enable) in the
0035  4156 ;           RDM Control Register.
0035  4157 ;       11. Start the clock by clearing the bits CLKCN1 and CLKCN0 in the DCSCR
0035  4158 ;           which will execute the micro-code to:
0035  4159 ;                1. Load RTEMP1 with the data from the WD register.
0035  4160 ;                2. Load the VA with zero.
0035  4161 ;                3. Write the model instruction from RTEMP1 to cache.
0035  4162 ;                4. Increment the VA for the next address.
0035  4163 ;                5. Write zeros into this cache location to prevent cache
0035  4164 ;                   parity errors from trapping the IRD1 decode.
0035  4165 ;                6. Zero the PC which will initiate instruction prefetch.
0035  4166 ;                7. Execute a BUT/IRD1TST to load the IR and OSR.
0035  4167 ;                8. Execute a NOP.
```

```
0035  4168 ;                9. Execute a BUT/IRDX.
0035  4169 ;               10. Halt.
0035  4170 ;
0035  4171 ;           NOTE! - The micro-code above cannot be single stepped.
0035  4172 ;
0035  4173 ;       12. Execute the micro-code for the second IRDX without uADINH asserted
0035  4174 ;           in order to trap the generated address.
0035  4175 ;       13. Execute a masked comparison of the control store address trace
0035  4176 ;           register with the expected CS address.
0035  4177 ;       14. If the two are different signal an error.
0035  4178 ;       15. Otherwise continue looping from step 5 to step 14 till finished.
0035  4179 ;       16. Execute the micro-code to place the CPU back into native mode
0035  4180 ;           and enable the console CTRL-P interrupt.
0035  4181 ;       17. Exit.
0035  4182 ;
0035  4183 ; DATA PATTERNS
0035  4184 ;
0035  4185 ;       The input data patterns were derived from chapter 4.4 of the CPU
0035  4186 ;       micro-sequencer specification dated on 9/25/78. Because operand
0035  4187 ;       specifiers are embedded into compatibility mode instructions, certain
0035  4188 ;       instructions have more than one opcode decode. Model opcodes were
0035  4189 ;       generated by first using Chart 4.4.1B to obtain the base opcodes for
0035  4190 ;       each compatibility mode instruction. These opcodes were then enhanced
0035  4191 ;       to include all the possible decodes. The enhancements were obtained
0035  4192 ;       by changing the operands to specify both register and non-register
0035  4193 ;       modes and by using Chart 4.4.1A to modify the operands to get all
0035  4194 ;       possible instruction decodes.
0035  4195 ;
0035  4196 ;       The expected control store addresses where obtained from the
0035  4197 ;       Compatibility Mode Ird Rom Definition in the CPU micro-code.
0035  4198 ;
0035  4199 ;       The following is a table that shows the instruction being decoded for
0035  4200 ;       a range of loop counts:
0035  4201 ;
0035  4202 ;       NOTE! - This table is NOT the same as that for CM IRD1.
0035  4203 ;
0035  4204 ;       Loop            Instruction             Loop            Instruction
0035  4205 ;
0035  4206 ;       01 - 10         JSR                     96              FSUB
0035  4207 ;       11 - 12         CLR                     97              FMUL
0035  4208 ;       13 - 14         COM                     98              FDIV
0035  4209 ;       15 - 16         INC                     99 - A0         (unused)
0035  4210 ;       17 - 18         DEC                     A1 - A2         SOB
0035  4211 ;       19 - 1A         NEG                     A3 - BE         (reserved)
0035  4212 ;       1B - 1C         ADC                     BF              BR
0035  4213 ;       1D - 1E         SBC                     C0              BNE
0035  4214 ;       1F - 20         TST                     C1              BEQ
0035  4215 ;       21 - 22         ROR                     C2              BGE
0035  4216 ;       23 - 24         ROL                     C3              BLT
0035  4217 ;       25 - 26         ASR                     C4              BGT
0035  4218 ;       27 - 28         ASL                     C5              BLE
0035  4219 ;       29 - 2A         MARK                    C6              BPL
0035  4220 ;       2B - 2C         MFPI                    C7              BMI
0035  4221 ;       2D - 2E         MTPI                    C8              BHI
0035  4222 ;       2F - 30         SXT                     C9              BLOS
0035  4223 ;       31 - 40         (unused)                CA              BVC
0035  4224 ;       41 - 48         EMT                     CB              BVS
```

```
0035  4225 ;       49 - 50         TRAP                    CC              BCC, BHIS
0035  4226 ;       51 - 52         CLRB                    CD              BCS, BLO
0035  4227 ;       53 - 54         COMB                    CE              HALT
0035  4228 ;       55 - 56         INCB                    CF              WAIT
0035  4229 ;       57 - 58         DECB                    D0              RTI
0035  4230 ;       59 - 5A         NEGB                    D1              BPT
0035  4231 ;       5B - 5C         ADCB                    D2              IOT
0035  4232 ;       5D - 5E         SBCB                    D3              RESET
0035  4233 ;       5F - 60         TSTB                    D4              RTT
0035  4234 ;       61 - 62         RORB                    D5              (unused)
0035  4235 ;       63 - 64         ROLB                    D6 - D9         MOV
0035  4236 ;       65 - 66         ASRB                    DA - DD         CMP
0035  4237 ;       67 - 68         ASLB                    DE - E1         BIT
0035  4238 ;       69 - 6A         (unused)                E2 - E5         BIC
0035  4239 ;       6B - 6C         MFPD                    E6 - E9         BIS
0035  4240 ;       6D - 6E         MTPD                    EA - ED         ADD
0035  4241 ;       6F - 80         (unused)
0035  4242 ;       81 - 84         MUL
0035  4243 ;       85 - 88         DIV
0035  4244 ;       89 - 8C         ASH
0035  4245 ;       8D - 90         ASHC
0035  4246 ;       91 - 94         XOR
0035  4247 ;       95              FADD
0035  4248 ;
0035  4249 ; LOGICAL DESCRIPTION
0035  4250 ;
0035  4251 ;       During an IRD1 in compatibility mode the CCS ROMs are fed addresses
0035  4252 ;       from the Compatibility Mode IRD1 and IRDX ROMs on the DPM. The DPM ROMs
0035  4253 ;       are addressed according to the following scheme:
0035  4254 ;
0035  4255 ;       Source          ROM Address Bits        Comment
0035  4256 ;
0035  4257 ;       CIR <7:0>       <10:3>                  Encoded opcode generated by the
0035  4258 ;                                               IRD gate array.
0035  4259 ;
0035  4260 ;       IRD1            <2>                     Distinguishes between the 1st
0035  4261 ;                                               operand and the 2nd operand or
0035  4262 ;                                               execution flows. 1 during IRD1
0035  4263 ;                                               0 otherwise.
0035  4264 ;
0035  4265 ;       IRD Counter <0> <1>                     Distinguishes between the 2nd
0035  4266 ;                                               operand and execution flows.
0035  4267 ;                                               0 when decoding OS2, 1 when
0035  4268 ;                                               transferring to execution flows
0035  4269 ;
0035  4270 ;       Register Mode   <0>                     1 if the operand being eval-
0035  4271 ;                                               uated specifies register mode.
0035  4272 ;                                               Generated by the IRD and PHB
0035  4273 ;                                               gate arrays.
0035  4274 ;
0035  4275 ;       In the case of a Compatibility Mode second IRDX, only execution flow
0035  4276 ;       addresses should be generated.
0035  4277 ;
0035  4278 ; ASSUMPTIONS
0035  4279 ;
0035  4280 ;       This test assumes the following logic has already been tested; the
0035  4281 ;       cache, XB's, the IR and the OSR.
```

```
                        4282 ;
                        4283 ; ERROR DESCRIPTION
                        4284 ;
                        4285 ;       Below is a table of possible error symptoms and their most likely
                        4286 ;       causes:
                        4287 ;
                        4288 ;               Symptom                                 Probable Cause
                        4289 ;       +-------------------------------+-------------------------------+
                        4290 ;       | The received control store    |                               |
                        4291 ;       | address is bad for one or sev-| One or several bad CM IRD1 and|
                        4292 ;       | eral instructions, or shows   | IRDX ROMs.                    |
                        4293 ;       | certain bits stuck at 1 or 0. |                               |
                        4294 ;       +-------------------------------+-------------------------------+
                        4295 ;       | The received control store    |                               |
                        4296 ;       | address is bad for every      |                               |
                        4297 ;       | instruction or branches to the| A bad IRD gate array.         |
                        4298 ;       | CM.BAD.IRD address in the     |                               |
                        4299 ;       | micro-code.                   |                               |
                        4300 ;       +-------------------------------+-------------------------------+
                        4301 ;       | Register mode fails to assert |                               |
                        4302 ;       | causing the wrong address to  | A bad IRD or PHB gate array.  |
                        4303 ;       | be generated.                 |                               |
                        4304 ;       +-------------------------------+-------------------------------+
                        4305 ;
                        4306 ;       On error this test will call out the IRD and PHB gate arrays and will
                        4307 ;       also type out the following data:
                        4308 ;
                        4309 ;       EXPECTED CONTROL STORE ADDRESS
                        4310 ;       RECEIVED CONTROL STORE ADDRESS
                        4311 ;       LOOP COUNT
                        4312 ;       DATA RESERVED FOR INSTRUCTION
                        4313 ;       OPCODE OF THE INSTRUCTION BEING DECODED
                        4314 ;
                        4315 ;--
                        4316 ;*****************************************************************************
                        4317 ;/////////////////////////////////////////////////////////////////////////////
                        4318
                        4319         INITIALIZE                                      ; INITIALIZE THE CPU
0037                    4320         FETCH           0                               ; CPU INTO COMP. MODE, MDR 0
003C                    4321         BRSTCLK         6                               ; AND DISABLE CONSOLE INTERRUPT
0040                    4322         EXPUTRAP                                        ; EXPECT NON-DCS ADDRESSES
0042                    4323         LOADREG         WDREG,1$,,L                     ; INITIALIZE THE WDREG
004A                    4324         LOOP            I,1,<^X0ED>                     ; LOOP FOR 239 INSTRUCTIONS
0051                    4325         LOADREG         WDREG,2$,I,W                    ; LOAD MODEL INSTRUCTION
0059                    4326         ERRLOOP                                         ; LOOP ON ERROR FROM HERE
005B                    4327         LOADREG         DCSCR,10$,,B                    ; KICK CPU OFF THE CS BUS
0063                    4328         LOADREG         DCSAD,11$,,B                    ; LOAD DCS ADDRESS COUNTER
006B                    4329         LOADREG         CSAMR,12$,,W                    ; PUT DCS OUT ON THE CS BUS
0073                    4330         LOADREG         RDCTRL,15$,,B                   ; PULSE TRAP STOP AND MASTER
007B                    4331         LOADREG         RDCTRL,13$,,B                   ; HALT ENABLE (TSTP, MSTP)
0083                    4332         LOADREG         DCSCR,14$,,B                    ; LET THE CLOCK RUN
008B                    4333         FETCH           <^X11>                          ; FETCH THE SECOND IRDX
0090                    4334         BRSTCLK         <^X13>,INH                      ; EXECUTE THE SECOND IRDX
0094                    4335         CMPREGMSK       CSTRP,3$,I,4$,,W                ; IS THE CS ADDRESS CORRECT?
00A0                    4336         IFERROR         2,<<IRD><PHB>>                  ; IF NOT, SIGNAL AN ERROR
00A7                    4337         ENDLOOP         I                               ; LOOP FOR ALL INSTRUCTIONS
00AA                    4338         FETCH           <^X14>                          ; PUT CPU INTO NATIVE MODE AND
```

```
                                00AF  4339          BRSTCLK         <^X19>                          ; ENABLE CONSOLE INTERRUPTS
                                00B3  4340          SKIP                                            ; END OF TEST
                                00BA  4341
                                00BA  4342 BEGIN_TEST_DATA
                                00BA
                                00BA          ;-----------------------------------------------------------------------
                                00BA  4343
                    00000000    00BA  4344 1$:      .LONG   0       ; LONGWORD TO INITIALIZE THE WD REGISTER
                                00BE  4345 ;
                                00BE  4346 ; Table of test opcodes.
                                00BE  4347 ;
                                00BE  4348 ;        The numbers to the right of the semicolons indicates the range
                                00BE  4349 ;        of the loop count for those patterns.
                                00BE  4350 ;
 08C0 0888 0880 0848 0840 0808 0800   00BE  4351 2$:      .WORD   ^X0800,^X0808,^X0840,^X0848,^X0880,^X0888,^X08C0 ; 01 - 07
 0988 0980 0948 0940 0908 0900 08C8   00CC  4352          .WORD   ^X08C8,^X0900,^X0908,^X0940,^X0948,^X0980,^X0988 ; 08 - 0E
 0A80 0A48 0A40 0A08 0A00 09C8 09C0   00DA  4353          .WORD   ^X09C0,^X09C8,^X0A00,^X0A08,^X0A40,^X0A48,^X0A80 ; 0F - 15
 0B48 0B40 0B08 0B00 0AC8 0AC0 0A88   00E8  4354          .WORD   ^X0A88,^X0AC0,^X0AC8,^X0B00,^X0B08,^X0B40,^X0B48 ; 16 - 1C
 0C40 0C08 0C00 0BC8 0BC0 0B88 0B80   00F6  4355          .WORD   ^X0B80,^X0B88,^X0BC0,^X0BC8,^X0C00,^X0C08,^X0C40 ; 1D - 23
 0D08 0D00 0CC8 0CC0 0C88 0C80 0C48   0104  4356          .WORD   ^X0C48,^X0C80,^X0C88,^X0CC0,^X0CC8,^X0D00,^X0D08 ; 24 - 2A
 0E00 0DC8 0DC0 0D88 0D80 0D48 0D40   0112  4357          .WORD   ^X0D40,^X0D48,^X0D80,^X0D88,^X0DC0,^X0DC8,^X0E00 ; 2B - 31
 0EC8 0EC0 0E88 0E80 0E48 0E40 0E08   0120  4358          .WORD   ^X0E08,^X0E40,^X0E48,^X0E80,^X0E88,^X0EC0,^X0EC8 ; 32 - 38
 0FC0 0F88 0F80 0F48 0F40 0F08 0F00   012E  4359          .WORD   ^X0F00,^X0F08,^X0F40,^X0F48,^X0F80,^X0F88,^X0FC0 ; 39 - 3F
 8888 8880 8848 8840 8808 8800 0FC8   013C  4360          .WORD   ^X0FC8,^X8800,^X8808,^X8840,^X8848,^X8880,^X8888 ; 40 - 46
 8980 8948 8940 8908 8900 88C8 88C0   014A  4361          .WORD   ^X88C0,^X88C8,^X8900,^X8908,^X8940,^X8948,^X8980 ; 47 - 4D
 8A48 8A40 8A08 8A00 89C8 89C0 8988   0158  4362          .WORD   ^X8988,^X89C0,^X89C8,^X8A00,^X8A08,^X8A40,^X8A48 ; 4E - 54
 8B40 8B08 8B00 8AC8 8AC0 8A88 8A80   0166  4363          .WORD   ^X8A80,^X8A88,^X8AC0,^X8AC8,^X8B00,^X8B08,^X8B40 ; 55 - 5B
 8C08 8C00 8BC8 8BC0 8B88 8B80 8B48   0174  4364          .WORD   ^X8B48,^X8B80,^X8B88,^X8BC0,^X8BC8,^X8C00,^X8C08 ; 5C - 62
 8D00 8CC8 8CC0 8C88 8C80 8C48 8C40   0182  4365          .WORD   ^X8C40,^X8C48,^X8C80,^X8C88,^X8CC0,^X8CC8,^X8D00 ; 63 - 69
 8DC8 8DC0 8D88 8D80 8D48 8D40 8D08   0190  4366          .WORD   ^X8D08,^X8D40,^X8D48,^X8D80,^X8D88,^X8DC0,^X8DC8 ; 6A - 70
 8EC0 8E88 8E80 8E48 8E40 8E08 8E00   019E  4367          .WORD   ^X8E00,^X8E08,^X8E40,^X8E48,^X8E80,^X8E88,^X8EC0 ; 71 - 77
 8F88 8F80 8F48 8F40 8F08 8F00 8EC8   01AC  4368          .WORD   ^X8EC8,^X8F00,^X8F08,^X8F40,^X8F48,^X8F80,^X8F88 ; 78 - 7E
 7200 7088 7080 7008 7000 8FC8 8FC0   01BA  4369          .WORD   ^X8FC0,^X8FC8,^X7000,^X7008,^X7080,^X7088,^X7200 ; 7F - 85
 7488 7480 7408 7400 7288 7280 7208   01C8  4370          .WORD   ^X7208,^X7280,^X7288,^X7400,^X7408,^X7480,^X7488 ; 86 - 8C
 7880 7808 7800 7688 7680 7608 7600   01D6  4371          .WORD   ^X7600,^X7608,^X7680,^X7688,^X7800,^X7808,^X7880 ; 8D - 93
 7A28 7A20 7A18 7A10 7A08 7A00 7888   01E4  4372          .WORD   ^X7888,^X7A00,^X7A08,^X7A10,^X7A18,^X7A20,^X7A28 ; 94 - 9A
 7E00 7CA8 7CA0 7C28 7C20 7AA8 7AA0   01F2  4373          .WORD   ^X7AA0,^X7AA8,^X7C20,^X7C28,^X7CA0,^X7CA8,^X7E00 ; 9B - A1
 F208 F200 F088 F080 F008 F000 7E80   0200  4374          .WORD   ^X7E80,^XF000,^XF008,^XF080,^XF088,^XF200,^XF208 ; A2 - A8
 F680 F608 F600 F480 F400 F288 F280   020E  4375          .WORD   ^XF280,^XF288,^XF400,^XF480,^XF600,^XF608,^XF680 ; A9 - AF
 FA08 FA00 F888 F880 F808 F800 F688   021C  4376          .WORD   ^XF688,^XF800,^XF808,^XF880,^XF888,^XFA00,^XFA08 ; B0 - B6
 FE00 FC88 FC80 FC08 FC00 FA88 FA80   022A  4377          .WORD   ^XFA80,^XFA88,^XFC00,^XFC08,^XFC80,^XFC88,^XFE00 ; B7 - BD
 0600 0500 0400 0300 0200 0100 FE80   0238  4378          .WORD   ^XFE80,^X0100,^X0200,^X0300,^X0400,^X0500,^X0600 ; BE - C4
 8500 8400 8300 8200 8100 8000 0700   0246  4379          .WORD   ^X0700,^X8000,^X8100,^X8200,^X8300,^X8400,^X8500 ; C5 - CB
 0004 0003 0002 0001 0000 8700 8600   0254  4380          .WORD   ^X8600,^X8700,^X0000,^X0001,^X0002,^X0003,^X0004 ; CC - D2
 1088 1080 1008 1000 0007 0006 0005   0262  4381          .WORD   ^X0005,^X0006,^X0007,^X1000,^X1008,^X1080,^X1088 ; D3 - D9
 3080 3008 3000 2088 2080 2008 2000   0270  4382          .WORD   ^X2000,^X2008,^X2080,^X2088,^X3000,^X3008,^X3080 ; DA - E0
 5008 5000 4088 4080 4008 4000 3088   027E  4383          .WORD   ^X3088,^X4000,^X4008,^X4080,^X4088,^X5000,^X5008 ; E1 - E7
      6088 6080 6008 6000 5088 5080   028C  4384          .WORD   ^X5080,^X5088,^X6000,^X6008,^X6080,^X6088         ; E8 - ED
                                0298  4385 ;
                                0298  4386 ; Table of control store addresses.
                                0298  4387 ;
                                0298  4388 ;        The numbers to the right of the semicolons indicates the range
                                0298  4389 ;        of the loop count for those patterns.
                                0298  4390 ;
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   0298  4391 3$:      .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 01 - 07
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   02A6  4392          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 08 - 0E
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   02B4  4393          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 0F - 15
```

```
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   02C2  4394          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 16 - 1C
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   02D0  4395          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 1D - 23
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   02DE  4396          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 24 - 2A
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   02EC  4397          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 2B - 31
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   02FA  4398          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 32 - 38
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   0308  4399          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 39 - 3F
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   0316  4400          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 40 - 46
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   0324  4401          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 47 - 4D
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   0332  4402          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 4E - 54
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   0340  4403          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 55 - 5B
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   034E  4404          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 5C - 62
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   035C  4405          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 63 - 69
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   036A  4406          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 6A - 70
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   0378  4407          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 71 - 77
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   0386  4408          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 78 - 7E
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   0394  4409          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 7F - 85
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   03A2  4410          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 86 - 8C
 03E2 03E2 03E2 03F8 03F8 03F8 03F8   03B0  4411          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03E2,^X03E2,^X03E2 ; 8D - 93
 03F8 03F8 03F8 03F8 03F8 03F8 03E2   03BE  4412          .WORD   ^X03E2,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 94 - 9A
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   03CC  4413          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 9B - A1
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   03DA  4414          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; A2 - A8
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   03E8  4415          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; A9 - AF
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   03F6  4416          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; B0 - B6
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   0404  4417          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; B7 - BD
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   0412  4418          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; BE - C4
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   0420  4419          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; C5 - CB
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   042E  4420          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; CC - D2
 03DC 03DC 03DC 03DC 03F8 03F8 03F8   043C  4421          .WORD   ^X03F8,^X03F8,^X03F8,^X03DC,^X03DC,^X03DC,^X03DC ; D3 - D9
 03E4 03E4 03E4 03DF 03DF 03DF 03DF   044A  4422          .WORD   ^X03DF,^X03DF,^X03DF,^X03DF,^X03E4,^X03E4,^X03E4 ; DA - E0
 03E3 03E3 03E5 03E5 03E5 03E5 03E4   0458  4423          .WORD   ^X03E4,^X03E5,^X03E5,^X03E5,^X03E5,^X03E3,^X03E3 ; E1 - E7
      03DD 03DD 03DD 03DD 03E3 03E3   0466  4424          .WORD   ^X03E3,^X03E3,^X03DD,^X03DD,^X03DD,^X03DD         ; E8 - ED
                                      0472  4425
                                3FFF  0472  4426 4$:      .WORD   ^X3FFF  ; MASK FOR CS ADDRESS COMPARISON
                                      0474  4427
                                  F8  0474  4428 10$:     .BYTE   ^X0F8   ; DCSCR  = uADINH!PARSTP!PARCHK!CLKCN1!CLKCN0
                                  07  0475  4429 11$:     .BYTE   ^X07    ; DCSAD  = 07
                                1807  0476  4430 12$:     .WORD   ^X1807  ; CSAMR  = 1807
                                  28  0478  4431 13$:     .BYTE   ^X28    ; RDCTRL = TSTP!MSTP
                                  98  0479  4432 14$:     .BYTE   ^X98    ; DCSCR  = uADINH!PARSTP!PARCHK
                                  00  047A  4433 15$:     .BYTE   ^X00    ; RDCTRL = 0
                                      047B  4434
                                      047B  4435 BEGIN_CPU_DATA
                                      0002  4436
                                      0002  4437 DCS_DATA
                                      0001  4438
U 00, 7700,C800,0364,0300,0470,1801   0001  4439 ;% 00   NOP                                    ; GET CPU UP TO SPEED
U 01, 7700,3800,3FFF,FFFF,8470,1802   000C  4443 ;% 01   LONLIT_[80000000]                      ; PLACES CPU INTO COMP. MODE
U 02, 7700,C800,5BE4,03D4,0070,1803   0017  4447 ;% 02   WB_R[LONLIT],CCPSL/PSL_WB.CCBR_ALUS   ; ...
U 03, 7700,0800,5BE4,0308,2470,1804   0022  4451 ;% 03   WCTRL/LOADCRAR,WB_R[TEMP2]            ; DISABLE CONSOLE INTERRUPT
U 04, 7700,4800,5BE4,030C,2C70,1805   002D  4455 ;% 04   WCTRL/CONWRITE,WB_R[TEMP3]            ; ...
U 05, 7700,C800,0364,0300,4E70,1806   0038  4459 ;% 05   MDR_0                                 ; ZERO THE MDR
U 06, 7300,C800,0364,0300,0470,1807   0043  4463 ;% 06   STOP.CLOCK                            ; HALT
                                      004E  4467
U 07, 7700,C800,0364,0300,0470,1808   004E  4468 ;% 07   NOP                                    ; GET THE CPU UP TO SPEED
U 08, 5700,4840,0364,0304,0470,1809   0059  4472 ;% 08   R[TEMP1]_RDM                           ; PUT MODEL OPCODE INTO RTEMP1
U 09, 7700,4800,5BE4,0314,4A70,180A   0064  4476 ;% 09   VA_R[TEMP5]                           ; ZERO THE VA
U 0A, 7701,C800,5BE4,0204,5590,180B   006F  4480 ;% 0A   WRITE.LONG R[TEMP1]                   ; WRITE OPCODE TO CACHE FOR XB0
```

```
U 0B. 7701,4800,5BE4,0204,55F0,180C  007A  4484 ;% 0B   WDR_UNROT(R[TEMP1]),BUS/PRB.RD,SIZE[LONG]
                                     0085  4488                                                  ; HOLD FOR SECOND CYCLE
U 0C. 7700,4800,0364,0300,4470,180D  0085  4489 ;% 0C   VA_VA+4                                  ; INCREMENT THE VA
U 0D. 7701,8800,5BE4,0214,5590,180E  0090  4493 ;% 0D   WRITE.LONG R[TEMP5]                      ; VALIDATE THE CACHE FOR XB1
U 0E. 7701,0800,5BE4,0214,55F0,180F  009B  4497 ;% 0E   WDR_UNROT(R[TEMP5]),BUS/PRB.RD,SIZE[LONG]
                                     00A6  4501                                                  ; HOLD FOR SECOND CYCLE
U 0F. 7700,4800,5BE4,0314,4870,1810  00A6  4502 ;% 0F   PC_R[TEMP5]                              ; INITIATE INSTRUCTION PREFETCH
U 10. 7700,4800,0364,1700,0470,1811  00B1  4506 ;% 10   BUT/IRD1TST                              ; LOAD THE IR AND OSR
U 11. 7700,4800,0364,0300,0470,1812  00BC  4510 ;% 11   NOP                                     ; NOP
U 12. 7700,8800,0364,0700,0470,1813  00C7  4514 ;% 12   BUT/IRDX                                ; EXECUTE AN IRDX
U 13. 7300,4800,0364,0300,0470,1814  00D2  4518 ;% 13   STOP.CLOCK                               ; HALT
                                     00DD  4522
U 14. 7700,C800,0364,0300,0470,1815  00DD  4523 ;% 14   NOP                                     ; GET CPU UP TO SPEED
U 15. 7700,0800,5BE4,0308,2470,1816  00E8  4527 ;% 15   WCTRL/LOADCRAR,WB_R[TEMP2]               ; ENABLE CONSOLE INTERRUPTS
U 16. 7700,0800,5BE4,0310,2C70,1817  00F3  4531 ;% 16   WCTRL/CONWRITE,WB_R[TEMP4]              ; ...
U 17. 7700,F800,7FFF,FFFF,8470,1818  00FE  4535 ;% 17   LONLIT_[0]                             ; PUT CPU BACK INTO NATIVE MODE
U 18. 7700,4800,5BE4,03D4,0070,1819  0109  4539 ;% 18   WB_R[LONLIT],CCPSL/PSL_WB.CCBR_ALUS    ; ...
U 19. 7300,C800,0364,0300,0470,181A  0114  4543 ;% 19   STOP.CLOCK                               ; HALT
                                     011F  4547
                                     011F  4548 RTEMP_DATA
                                     0001  4549
                           00000000  0001  4550          .LONG   0               ; RTEMP0 = RESERVED FOR INSTRUCTION USE
                           00000000  0005  4551          .LONG   0               ; RTEMP1 = INSTRUCTION OPCODE
                           00C00000  0009  4552          .LONG   ^X00C00000      ; RTEMP2 = CRAR RECV STATUS
                           00400000  000D  4553          .LONG   ^X00400000      ; RTEMP3 = DISABLES INTERRUPT
                           00900000  0011  4554          .LONG   ^X00900000      ; RTEMP4 = ENABLES INTERRUPT
                           00000000  0015  4555          .LONG   0               ; RTEMP5 = INSTRUCTION ADDRESS
                                     0019  4556
                                     0019  4557 END_CPU_DATA
                                     047B  4558
                                     047B  4559 END_TEST_DATA
                                     047B
                                     047B       ;-----------------------------------------------------------------------------
                                     047B  4560 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
0035        .SBTTL  "TEST   0DB     CM Second IRDX Instruction Address Decode Test 2
0035  4562 ;*******************************************************************************
0035  4563 ;++
0035  4564 ;
0035  4565 ; FUNCTIONAL DESCRIPTION
0035  4566 ;
0035  4567 ;       This test makes sure the proper control store addresses are generated
0035  4568 ;       for the second compatiblity mode IRDX. The test is implemented by using
0035  4569 ;       two data tables. The first table consists of assembled VAX compatibil-
0035  4570 ;       ity mode macro instructions. The second table consists of expected
0035  4571 ;       control store addresses for the second IRDX. This test decodes those
0035  4572 ;       instructions not done in test one due to insufficient RDM memory space.
0035  4573 ;
0035  4574 ; TEST ALGORITHM
0035  4575 ;
0035  4576 ;       The IRDX micro-order will assert micro-addresses onto the CS address
0035  4577 ;       bus which will cause the RDM to stop the processor clock. Thus if the
0035  4578 ;       first IRDX is executed using the FETCH and BRSTCLK pseudo-ops, the
0035  4579 ;       clock will stop before the IRDX is finished. This causes bad addresses
0035  4580 ;       to be generated for some of the macro instructions being decoded. To
0035  4581 ;       prevent this from happening, the LOADREG pseudo-op is used to execute
0035  4582 ;       the micro-code that fetches the macro instructions, loads the IR and
0035  4583 ;       OSR and does the first IRDX. The second IRDX can be executed using the
0035  4584 ;       FETCH and BRSTCLK pseudo-ops since the address that is generated will
0035  4585 ;       not depend on whether that micro-order finishes execution.
0035  4586 ;
0035  4587 ;       This test is executed in the following manner:
0035  4588 ;
0035  4589 ;        1. Initialize the CPU.
0035  4590 ;        2. Execute micro-code to place the CPU into compatibility mode,
0035  4591 ;           disable the console CTRL-P interrupt and zeroes the MDR in order
0035  4592 ;           to allow the IRDX to function.
0035  4593 ;        3. Initialize the WD register with zero.
0035  4594 ;        4. Prepare a loop for 49 instructions.
0035  4595 ;        5. Load the WD register with a model opcode from the opcode table
0035  4596 ;           indexed by the loop count.
0035  4597 ;        6. Take the CPU off the CS address bus and hold the CPU clock
0035  4598 ;           off by setting the bits uADINH, CLKCN1, CLKCN0, PARSTP, and PARCHK
0035  4599 ;           in the DCS Control Register (DCSCR).
0035  4600 ;        7. Load the DCS Address Counter (DCSAD) with the starting location of
0035  4601 ;           the test micro-code.
0035  4602 ;        8. Load the Control Store Address Match Register (CSAMR) with the
0035  4603 ;           starting address of the test micro-code to enable DCS onto the
0035  4604 ;           CPU control store bus.
0035  4605 ;        9. Initialize the RDM Control Register (RDCTRL) with zero.
0035  4606 ;       10. Set the bits TSTP (trap stop enable) and MSTP (master halt enable)
0035  4607 ;           in the RDM Control Register.
0035  4608 ;       11. Start the clock by clearing the bits CLKCN1 and CLKCN0 in the DCSCR
0035  4609 ;           which will execute the micro-code to:
0035  4610 ;                1. Load RTEMP1 with the data from the WD register.
0035  4611 ;                2. Load the VA with zero.
0035  4612 ;                3. Write the model instruction from RTEMP1 to cache.
0035  4613 ;                4. Increment the VA for the next address.
0035  4614 ;                5. Write zeros into this cache location to prevent cache
0035  4615 ;                   parity errors from trapping the IRD1 decode.
0035  4616 ;                6. Zero the PC which will initiate instruction prefetch.
0035  4617 ;                7. Execute a BUT/IRD1TST to load the IR and OSR.
```

```
0035  4618 ;                8. Execute a NOP.
0035  4619 ;                9. Execute a BUT/IRDX.
0035  4620 ;               10. Halt.
0035  4621 ;
0035  4622 ;           NOTE! - The micro-code above cannot be single stepped.
0035  4623 ;
0035  4624 ;       12. Execute the micro-code for the second IRDX without uADINH asserted
0035  4625 ;           in order to trap the generated address.
0035  4626 ;       13. Execute a masked comparison of the control store address trace
0035  4627 ;           register with the expected CS address.
0035  4628 ;       14. If the two are different signal an error.
0035  4629 ;       15. Otherwise continue looping from step 5 to step 14 till finished.
0035  4630 ;       16. Execute the micro-code to place the CPU back into native mode
0035  4631 ;           and enable the console CTRL-P interrupt.
0035  4632 ;       17. Exit.
0035  4633 ;
0035  4634 ; DATA PATTERNS
0035  4635 ;
0035  4636 ;       The input data patterns were derived from chapter 4.4 of the CPU
0035  4637 ;       micro-sequencer specification dated on 9/25/78. Because operand
0035  4638 ;       specifiers are embedded into compatibility mode instructions, certain
0035  4639 ;       instructions have more than one opcode decode. Model opcodes were
0035  4640 ;       generated by first using Chart 4.4.1B to obtain the base opcodes for
0035  4641 ;       each compatibility mode instruction. These opcodes were then enhanced
0035  4642 ;       to include all the possible decodes. The enhancements were obtained
0035  4643 ;       by changing the operands to specify both register and non-register
0035  4644 ;       modes and by using Chart 4.4.1A to modify the operands to get all
0035  4645 ;       possible instruction decodes.
0035  4646 ;
0035  4647 ;       The expected control store addresses where obtained from the
0035  4648 ;       Compatibility Mode Ird Rom Definition in the CPU micro-code.
0035  4649 ;
0035  4650 ;       The following is a table that shows the instruction being decoded for
0035  4651 ;       a range of loop counts:
0035  4652 ;
0035  4653 ;       NOTE! - This table is NOT the same as that for CM IRD1.
0035  4654 ;
0035  4655 ;       Loop            Instruction
0035  4656 ;
0035  4657 ;       01 - 04         MOVB
0035  4658 ;       05 - 08         CMPB
0035  4659 ;       09 - 0C         BITB
0035  4660 ;       0D - 10         BICB
0035  4661 ;       11              BISB
0035  4662 ;       12 - 15         SUB
0035  4663 ;       16 - 1D         JMP
0035  4664 ;       1E              RTS
0035  4665 ;       1F - 20         (unused)
0035  4666 ;       21              SPL
0035  4667 ;       22              NOP
0035  4668 ;       23 - 26         (reserved)
0035  4669 ;       27 - 31         SWAB
0035  4670 ;
0035  4671 ; LOGICAL DESCRIPTION
0035  4672 ;
0035  4673 ;       During an IRD1 in compatibility mode the CCS ROMs are fed addresses
0035  4674 ;       from the Compatibility Mode IRD1 and IRDX ROMs on the DPM. The DPM ROMs
```

```
0035  4675 ;       are addressed according to the following scheme:
0035  4676 ;
0035  4677 ;       Source          ROM Address Bits        Comment
0035  4678 ;
0035  4679 ;       CIR <7:0>       <10:3>                  Encoded opcode generated by the
0035  4680 ;                                               IRD gate array.
0035  4681 ;
0035  4682 ;       IRD1            <2>                     Distinguishes between the 1st
0035  4683 ;                                               operand and the 2nd operand or
0035  4684 ;                                               execution flows. 1 during IRD1
0035  4685 ;                                               0 otherwise.
0035  4686 ;
0035  4687 ;       IRD Counter <0> <1>                     Distinguishes between the 2nd
0035  4688 ;                                               operand and execution flows.
0035  4689 ;                                               0 when decoding OS2, 1 when
0035  4690 ;                                               transferring to execution flows
0035  4691 ;
0035  4692 ;       Register Mode   <0>                     1 if the operand being eval-
0035  4693 ;                                               uated specifies register mode.
0035  4694 ;                                               Generated by the IRD and PHB
0035  4695 ;                                               gate arrays.
0035  4696 ;
0035  4697 ;       In the case of a Compatibility Mode second IRDX, only execution flow
0035  4698 ;       addresses should be generated.
0035  4699 ;
0035  4700 ; ASSUMPTIONS
0035  4701 ;
0035  4702 ;       This test assumes the following logic has already been tested; the
0035  4703 ;       cache, XB's, the IR and the OSR.
0035  4704 ;
0035  4705 ; ERROR DESCRIPTION
0035  4706 ;
0035  4707 ;       Below is a table of possible error symptoms and their most likely
0035  4708 ;       causes:
0035  4709 ;
0035  4710 ;               Symptom                                 Probable Cause
0035  4711 ;       +-------------------------------+-------------------------------+
0035  4712 ;       | The received control store    |                               |
0035  4713 ;       | address is bad for one or sev-| One or several bad CM IRD1 and|
0035  4714 ;       | eral instructions, or shows   | IRDX ROMs.                    |
0035  4715 ;       | certain bits stuck at 1 or 0. |                               |
0035  4716 ;       +-------------------------------+-------------------------------+
0035  4717 ;       | The received control store    |                               |
0035  4718 ;       | address is bad for every      |                               |
0035  4719 ;       | instruction or branches to the| A bad IRD gate array.         |
0035  4720 ;       | CM.BAD.IRD address in the     |                               |
0035  4721 ;       | micro-code.                   |                               |
0035  4722 ;       +-------------------------------+-------------------------------+
0035  4723 ;       | Register mode fails to assert |                               |
0035  4724 ;       | causing the wrong address to  | A bad IRD or PHB gate array.  |
0035  4725 ;       | be generated.                 |                               |
0035  4726 ;       +-------------------------------+-------------------------------+
0035  4727 ;
0035  4728 ;       On error this test will call out the IRD and PHB gate arrays and will
0035  4729 ;       also type out the following data:
0035  4730 ;
0035  4731 ;       EXPECTED CONTROL STORE ADDRESS
```

```
                                0035  4732 ;       RECEIVED CONTROL STORE ADDRESS
                                0035  4733 ;       LOOP COUNT
                                0035  4734 ;       DATA RESERVED FOR INSTRUCTION
                                0035  4735 ;       OPCODE OF THE INSTRUCTION BEING DECODED
                                0035  4736 ;
                                0035  4737 ;--
                                0035  4738 ;*******************************************************************************************
                                0035  4739 ;///////////////////////////////////////////////////////////////////////////////////////
                                0035  4740
                                0035  4741         INITIALIZE                                      ; INITIALIZE THE CPU
                                0037  4742         FETCH           0                               ; CPU INTO COMP. MODE, MDR_0
                                003C  4743         BRSTCLK         6                               ; AND DISABLE CONSOLE INTERRUPT
                                0040  4744         EXPUTRAP                                        ; EXPECT NON-DCS ADDRESSES
                                0042  4745         LOADREG         WDREG,1$,,L                     ; INITIALIZE THE WD REGISTER
                                004A  4746         LOOP            I,1,<^X31>                      ; LOOP FOR 49 INSTRUCTIONS
                                0051  4747         LOADREG         WDREG,2$,I,W                    ; LOAD MODEL INSTRUCTION
                                0059  4748         ERRLOOP                                         ; LOOP ON ERROR FROM HERE
                                005B  4749         LOADREG         DCSCR,10$,,B                    ; KICK CPU OFF THE CS BUS
                                0063  4750         LOADREG         DCSAD,11$,,B                    ; LOAD DCS ADDRESS COUNTER
                                006B  4751         LOADREG         CSAMR,12$,,W                    ; PUT DCS OUT ON THE CS BUS
                                0073  4752         LOADREG         RDCTRL,15$,,B                   ; PULSE TRAP STOP AND MASTER
                                007B  4753         LOADREG         RDCTRL,13$,,B                   ; HALT ENABLE (TSTP, MSTP)
                                0083  4754         LOADREG         DCSCR,14$,,B                    ; LET THE CLOCK RUN
                                008B  4755         FETCH           <^X11>                          ; FETCH THE SECOND IRDX
                                0090  4756         BRSTCLK         <^X13>,INH                      ; EXECUTE THE SECOND IRDX
                                0094  4757         CMPREGMSK       CSTRP,3$,I,4$,,W                ; IS THE CS ADDRESS CORRECT?
                                00A0  4758         IFERROR         2,<<IRD><PHB>>                  ; IF NOT, SIGNAL AN ERROR
                                00A7  4759         ENDLOOP         I                               ; LOOP FOR ALL INSTRUCTIONS
                                00AA  4760         FETCH           <^X14>                          ; PUT CPU INTO NATIVE MODE AND
                                00AF  4761         BRSTCLK         <^X19>                          ; ENABLE CONSOLE INTERRUPTS
                                00B3  4762         SKIP                                            ; END OF TEST
                                00BA  4763
                                00BA  4764 BEGIN_TEST_DATA
                                00BA
                                00BA       ;-------------------------------------------------------------------------
                                00BA  4765
                    00000000    00BA  4766 1$:      .LONG   0       ; LONGWORD USED TO INITIALIZE THE WD REGISTER
                                00BE  4767 ;
                                00BE  4768 ; Table of test opcodes.
                                00BE  4769 ;
                                00BE  4770 ;       The numbers to the right of the semicolons indicates the range
                                00BE  4771 ;       of the loop count for those patterns.
                                00BE  4772 ;
A080 A008 A000 9088 9080 9008 9000  00BE  4773 2$:      .WORD   ^X9000,^X9008,^X9080,^X9088,^XA000,^XA008,^XA080 ; 01 - 07
C008 C000 B088 B080 B008 B000 A088  00CC  4774          .WORD   ^XA088,^XB000,^XB008,^XB080,^XB088,^XC000,^XC008 ; 08 - 0E
E000 D088 D080 D008 D000 C088 C080  00DA  4775          .WORD   ^XC080,^XC088,^XD000,^XD008,^XD080,^XD088,^XE000 ; 0F - 15
0058 0050 0048 0040 E088 E080 E008  00E8  4776          .WORD   ^XE008,^XE080,^XE088,^X0040,^X0048,^X0050,^X0058 ; 16 - 1C
0090 0088 0080 0078 0070 0068 0060  00F6  4777          .WORD   ^X0060,^X0068,^X0070,^X0078,^X0080,^X0088,^X0090 ; 1D - 23
00C0 00B8 00B0 00A9 00A1 00A0 0098  0104  4778          .WORD   ^X0098,^X00A0,^X00A1,^X00A9,^X00B0,^X00B8,^X00C0 ; 24 - 2A
00F8 00F0 00E8 00E0 00D8 00D0 00C8  0112  4779          .WORD   ^X00C8,^X00D0,^X00D8,^X00E0,^X00E8,^X00F0,^X00F8 ; 2B - 31
                                0120  4780 ;
                                0120  4781 ; Table of control store addresses.
                                0120  4782 ;
                                0120  4783 ;       The numbers to the right of the semicolons indicates the range
                                0120  4784 ;       of the loop count for those patterns.
                                0120  4785 ;
03DF 03DF 03DF 03DC 03DC 03DC 03DC  0120  4786 3$:      .WORD   ^X03DC,^X03DC,^X03DC,^X03DC,^X03DF,^X03DF,^X03DF ; 01 - 07
```

```
 03E5 03E5 03E4 03E4 03E4 03E4 03DF   012E  4787          .WORD   ^X03DF,^X03E4,^X03E4,^X03E4,^X03E4,^X03E5,^X03E5 ; 08 - 0E
 03DE 03E3 03E3 03E3 03E3 03E5 03E5   013C  4788          .WORD   ^X03E5,^X03E5,^X03E3,^X03E3,^X03E3,^X03E3,^X03DE ; 0F - 15
 03F8 03F8 03F8 03F8 03DE 03DE 03DE   014A  4789          .WORD   ^X03DE,^X03DE,^X03DE,^X03F8,^X03F8,^X03F8,^X03F8 ; 16 - 1C
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   0158  4790          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 1D - 23
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   0166  4791          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 24 - 2A
 03F8 03F8 03F8 03F8 03F8 03F8 03F8   0174  4792          .WORD   ^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8,^X03F8 ; 2B - 31
                                      0182  4793
                                3FFF  0182  4794 4$:      .WORD   ^X3FFF  ; MASK FOR CS ADDRESS COMPARISON
                                      0184  4795
                                  F8  0184  4796 10$:     .BYTE   ^X0F8   ; DCSCR  = uADINH!PARSTP!PAR_CHK!CLKCN1!CLKCN0
                                  07  0185  4797 11$:     .BYTE   ^X07    ; DCSAD  = 07
                                1807  0186  4798 12$:     .WORD   ^X1807  ; CSAMR  = 1807
                                  28  0188  4799 13$:     .BYTE   ^X28    ; RDCTRL = TSTP!MSTP
                                  98  0189  4800 14$:     .BYTE   ^X98    ; DCSCR  = uADINH!PARSTP!PARCHK
                                  00  018A  4801 15$:     .BYTE   ^X00    ; RDCTRL = 0
                                      018B  4802
                                      018B  4803 BEGIN_CPU_DATA
                                      0002  4804
                                      0002  4805 DCS_DATA
                                      0001  4806
U 00, 7700,C800,0364,0300,0470,1801   0001  4807 ;% 00   NOP                                     ; GET CPU UP TO SPEED
U 01, 7700,3800,3FFF,FFFF,8470,1802   000C  4811 ;% 01   LONLIT_[80000000]                       ; PLACES CPU INTO COMP. MODE
U 02, 7700,C800,5BE4,03D4,0070,1803   0017  4815 ;% 02   WB_R[LONLIT],CCPSL/PSL_WB.CCBR_ALUS    ; ...
U 03, 7700,0800,5BE4,0308,2470,1804   0022  4819 ;% 03   WCTRL/LOADCRAR,WB_R[TEMP2]              ; DISABLE CONSOLE INTERRUPT
U 04, 7700,4800,5BE4,030C,2C70,1805   002D  4823 ;% 04   WCTRL/CONWRITE,WB_R[TEMP3]             ; ...
U 05, 7700,C800,0364,0300,4E70,1806   0038  4827 ;% 05   MDR_0                                  ; ZERO THE MDR
U 06, 7300,C800,0364,0300,0470,1807   0043  4831 ;% 06   STOP.CLOCK                             ; HALT
                                      004E  4835
U 07, 7700,C800,0364,0300,0470,1808   004E  4836 ;% 07   NOP                                     ; GET THE CPU UP TO SPEED
U 08, 5700,4840,0364,0304,0470,1809   0059  4840 ;% 08   R[TEMP1]_RDM                            ; PUT MODEL OPCODE INTO RTEMP1
U 09, 7700,4800,5BE4,0314,4A70,180A   0064  4844 ;% 09   VA_R[TEMP5]                            ; ZERO THE VA
U 0A, 7701,C800,5BE4,0204,5590,180B   006F  4848 ;% 0A   WRITE.LONG R[TEMP1]                    ; WRITE OPCODE TO CACHE FOR XB0
U 0B, 7701,4800,5BE4,0204,55F0,180C   007A  4852 ;% 0B   WDR_UNROT(R[TEMP1]),BUS/PRB.RD,SIZE[LONG]
                                      0085  4856                                              ; HOLD FOR SECOND CYCLE
U 0C, 7700,4800,0364,0300,4470,180D   0085  4857 ;% 0C   VA_VA+4                               ; INCREMENT THE VA
U 0D, 7701,8800,5BE4,0214,5590,180E   0090  4861 ;% 0D   WRITE.LONG R[TEMP5]                    ; VALIDATE THE CACHE FOR XB1
U 0E, 7701,0800,5BE4,0214,55F0,180F   009B  4865 ;% 0E   WDR_UNROT(R[TEMP5]),BUS/PRB.RD,SIZE[LONG]
                                      00A6  4869                                              ; HOLD FOR SECOND CYCLE
U 0F, 7700,4800,5BE4,0314,4870,1810   00A6  4870 ;% 0F   PC_R[TEMP5]                            ; INITIATE INSTRUCTION PREFETCH
U 10, 7700,4800,0364,1700,0470,1811   00B1  4874 ;% 10   BUT/IRD1TST                           ; LOAD THE IR AND OSR
U 11, 7700,4800,0364,0300,0470,1812   00BC  4878 ;% 11   NOP                                     ; NOP
U 12, 7700,8800,0364,0700,0470,1813   00C7  4882 ;% 12   BUT/IRDX                              ; EXECUTE AND IRDX
U 13, 7300,4800,0364,0300,0470,1814   00D2  4886 ;% 13   STOP.CLOCK                             ; HALT
                                      00DD  4890
U 14, 7700,C800,0364,0300,0470,1815   00DD  4891 ;% 14   NOP                                     ; GET CPU UP TO SPEED
U 15, 7700,0800,5BE4,0308,2470,1816   00E8  4895 ;% 15   WCTRL/LOADCRAR,WB_R[TEMP2]              ; ENABLE CONSOLE INTERRUPTS
U 16, 7700,0800,5BE4,0310,2C70,1817   00F3  4899 ;% 16   WCTRL/CONWRITE,WB_R[TEMP4]             ; ...
U 17, 7700,F800,7FFF,FFFF,8470,1818   00FE  4903 ;% 17   LONLIT_[0]                             ; PUT CPU BACK INTO NATIVE MODE
U 18, 7700,4800,5BE4,03D4,0070,1819   0109  4907 ;% 18   WB_R[LONLIT],CCPSL/PSL_WB.CCBR_ALUS    ; ...
U 19, 7300,C800,0364,0300,0470,181A   0114  4911 ;% 19   STOP.CLOCK                             ; HALT
                                      011F  4915
                                      011F  4916 RTEMP_DATA
                                      0001  4917
                          00000000    0001  4918          .LONG   0               ; RTEMP0 = RESERVED FOR INSTRUCTION USE
                          00000000    0005  4919          .LONG   0               ; RTEMP1 = INSTRUCTION OPCODE
                          00C00000    0009  4920          .LONG   ^X00C00000      ; RTEMP2 = CRAR RECV STATUS
                          00400000    000D  4921          .LONG   ^X00400000      ; RTEMP3 = DISABLES INTERRUPT
```

```
00900000  0011  4922          .LONG   ^X00900000      ; RTEMP4 = ENABLES INTERRUPT
00000000  0015  4923          .LONG   0               ; RTEMP5 = INSTRUCTION ADDRESS
          0019  4924
          0019  4925 END_CPU_DATA
          018B  4926
          018B  4927 END_TEST_DATA
          018B
          018B        ;-------------------------------------------------------------------------
          018B  4928 ENDTEST
```

```
                        .SBTTL  "TEST   0DC     TEST CCS FOR PARITY ERRORS
001F   4930 ;*******************************************************************************
001F   4931 ;++
001F   4932 ;
001F   4933 ; FUNCTIONAL DESCRIPTION:
001F   4934 ;
001F   4935 ;       THIS TEST ADDRESSES ALL THE CPU MICRO CODE IN AN EFFORT TO DETECT
001F   4936 ;       PARITY ERRORS.
001F   4937 ;
001F   4938 ;
001F   4939 ; TEST ALGORITHM:
001F   4940 ;
001F   4941 ;       1.  CREATE A MICRO WORD WITH BAD PARITY
001F   4942 ;       2.  LOAD THIS MICRO WORD INTO DCS ADDRESS 0
001F   4943 ;       3.  INSURE THE CPU CLOCK IS OFF
001F   4944 ;       4.  SET THE DCS ADDRESS TO 0
001F   4945 ;       5.  LOAD ZERO INTO THE CSAMR REGISTER
001F   4946 ;       6.  SET THE PARCHK BIT TO ALLOW THE CSAMR TO DRIVE THE CS ADDRESS
001F   4947 ;       7.  CLEAR MASTER HALT ENABLE TO FREE THE RDM CLOCK LOGIC
001F   4948 ;       8.  SET DC LOW TO FORCE THE CPU CLOCK TO RUN
001F   4949 ;       9.  PERFORM WAIT LOOP OF 10
001F   4950 ;      10.  CLEAR DC LOW
001F   4951 ;      11.  TEST THE CS ADDRESS FOR 17FF
001F   4952 ;      12.  IF NO ERROR INIT THE CPU AND GO TO NEXT TEST
001F   4953 ;
001F   4954 ;
001F   4955 ; TEST PATTERNS:
001F   4956 ;
001F   4957 ;       NONE
001F   4958 ;
001F   4959 ;
001F   4960 ; LOGIC DESCRIPTION:
001F   4961 ;
001F   4962 ;       THIS TEST SIMPLY ADDRESSES ALL THE CPU MICRO CODE AND WATCHES FOR
001F   4963 ;       A PARITY ERROR.  WHILE LOOPING ON THIS TEST, MANUFACTURING CAN
001F   4964 ;       ENVIRONMENTALLY STRESS THE CPU CCS MODULE AND DETECT MARGINAL
001F   4965 ;       PARITY ERROR CONDITIONS.  THE CPU CONTROL STORE HAS A PARITY ERROR
001F   4966 ;       BLASTED INTO LOCATION 17FD.  IF THIS LOCATION IS BAD THEN THE CLOCK
001F   4967 ;       WILL STOP AT DCS ADDRESS 0 WHERE WE LOADED AN ADDITIONAL PARITY ERROR.
001F   4968 ;
001F   4969 ;
001F   4970 ; ASSUMPTIONS:
001F   4971 ;
001F   4972 ;       THIS TEST ASSUMES THE DPM PARITY CHECKING LOGIC IS OPERATIONAL.
001F   4973 ;
001F   4974 ;
001F   4975 ; ERROR DESCRIPTION:
001F   4976 ;
001F   4977 ;       EXPECTED CS ADDRESS
001F   4978 ;       RECEIVED CS ADDRESS
001F   4979 ;
001F   4980 ;       NOTE:  THE RECEIVED ADDRESS WILL BE TWO (2) AHEAD OF THE ACTUAL
001F   4981 ;                ADDRESS WITH THE PARITY ERROR.
001F   4982 ;
001F   4983 ;--
001F   4984 ;*******************************************************************************
001F   4985
```

```
                        001F  4986
                        001F  4987          INITIALIZE                                  ;INIT THE CPU
                        0021  4988          LDFIELD         2$,,B,11$,78,78,,,,NP,      ;CREATE BAD PARITY
                        0032  4989          LOADDCS         11$,,0,1                    ;LOAD MICRO WORD INTO DCS
                        0039  4990          LOADREG         DCSCR,1$,,B                 ;STOP THE CPU CLOCK
                        0041  4991          LOADREG         DCSAD,2$,,B                 ;SET DCS ADDRESS TO ZERO
                        0049  4992          LOADREG         CSAMR,4$,,W                 ;LOAD ADDRESS 0
                        0051  4993          LOADREG         DCSCR,5$,,B                 ;SET PARCHK BIT
                        0059  4994          LOADREG         RDCTRL,2$,,B                ;CLEAR MASTER HALT ENABLE
                        0061  4995          LOADREG         RDCTRL,3$,,B                ;SET MASTER HALT ENABLE & DC LO
                        0069  4996          LOOP            I,1,10                      ;ALLOW MICRO CODE TO BE READ
                        0070  4997          ENDLOOP         I                           ;...
                        0073  4998          LOADREG         RDCTRL,2$,,B                ;CLEAR MASTER HALT AND DC LO
                        007B  4999          CMPREGMSK       CSBUF,8$,,9$,,W,            ;TEST FOR ADDRESS 17FF
                        0087  5000          IFERROR         ,,<CCS>                     ;PARITY ERROR
                        008D  5001          INITIALIZE                                  ;CLEAR ANY PARITY ERRORS
                        008F  5002          SKIP                                        ;GO TO NEXT TEST
                        0096  5003
                        0096  5004
                        0096  5005 BEGIN_TEST_DATA
                        0096
                        0096          ;-------------------------------------------------------------------------
                        0096  5006
                  E0    0096  5007 1$:      .BYTE   ^XE0                                ;STOP CLOCK AND UADDR INH
                  00    0097  5008 2$:      .BYTE   ^X00                                ;CLEAR MASTER HALT ENABLE
                  48    0098  5009 3$:      .BYTE   ^X48                                ;SET MASTER HALT ENABLE
                0000    0099  5010 4$:      .WORD   ^X0000                              ;STARTING ADDRESS IN CCS
                  F8    009B  5011 5$:      .BYTE   ^XF8                                ;SET PARCHK BIT
                17FF    009C  5012 8$:      .WORD   ^X17FF                              ;ADDRESS OF SUCCESSFUL HALT
                3FFF    009E  5013 9$:      .WORD   ^X3FFF                              ;MASK FOR CMPREGMSK PSEUDO OP
                  A8    00A0  5014 10$:     .BYTE   ^XA8                                ;SINGLE STEP THE CLOCK
                        00A1  5015 11$:
U 00, 7700,C800,0364,0300,0470,1801   00A1  5016 ;% 00    NOP                           ;MICRO WORD FOR DCS
                        00AC  5020
                        00AC  5021 END_TEST_DATA
                        00AC
                        00AC          ;-------------------------------------------------------------------------
                        00AC  5022 ENDTEST
```

```
                        001A        .SBTTL  "TEST   ODD     RSRC/ZERO TIMING TEST
                        001A  5024 ;*******************************************************************************
                        001A  5025 ;++
                        001A  5026 ;
                        001A  5027 ; FUNCTIONAL DESCRIPTION:
                        001A  5028 ;
                        001A  5029 ;       THIS TEST ATTEMPTS TO DETECT SRM GATE ARRAYS WHICH MAY HAVE IMPROPER
                        001A  5030 ;       RESISTANCE VALUES ON THE RBUS.
                        001A  5031 ;
                        001A  5032 ;
                        001A  5033 ; TEST ALGORITHM:
                        001A  5034 ;
                        001A  5035 ;       1.  SET UP MICRO CODE TO HAVE CLKX OFF
                        001A  5036 ;       2.  CREATE A TEST LOOP OF 4000
                        001A  5037 ;       3.  EXECUTE DCS PROGRAM
                        001A  5038 ;               a.  SET-UP MDR TO ONE'S
                        001A  5039 ;               b.  CONDITION RBUS TO ALL ONE'S
                        001A  5040 ;               c.  HAVE SRM PULL RBUS TO ALL ZERO'S AND BRANCH ON SRK STATUS
                        001A  5041 ;       4.  TEST CS ADDRESS FOR CORRECT BRANCH ADDRESS
                        001A  5042 ;       5.  IF NO ERROR GO TO STEP 2 UNTIL TEST LOOP IS EXHAUSTED
                        001A  5043 ;
                        001A  5044 ;
                        001A  5045 ; TEST PATTERNS:
                        001A  5046 ;
                        001A  5047 ;       NONE
                        001A  5048 ;
                        001A  5049 ;
                        001A  5050 ; LOGIC DESCRIPTION:
                        001A  5051 ;
                        001A  5052 ;       THE INTENT OF THIS TEST IS TO CREATE A WORSE CASE CONDITION IN WHICH
                        001A  5053 ;       THE SRM'S ARE FORCED TO PULL THE RBUS TO ZERO (LOGICAL HIGH).  IF THE
                        001A  5054 ;       RESISTANCE OF THE INTERNAL SRM PULL-UP RESISTERS IS TOO HIGH THE RBUS
                        001A  5055 ;       DATA WILL NOT SET-UP IN TIME AND THE TEST WILL FAIL.  THE ONLY FIX IS
                        001A  5056 ;       TO REPLACE ALL THE SRM'S AND RERUN THE TEST.
                        001A  5057 ;
                        001A  5058 ;
                        001A  5059 ; ASSUMPTIONS:
                        001A  5060 ;
                        001A  5061 ;       THIS TEST ASSUMES THAT ALL OTHER MICRODIAGNOSTCS HAVE PASSED
                        001A  5062 ;       (MIC AND DPM).
                        001A  5063 ;
                        001A  5064 ;
                        001A  5065 ; ERROR DESCRIPTION:
                        001A  5066 ;
                        001A  5067 ;       EXPECTED CS ADDRESS
                        001A  5068 ;       RECEIVED CS ADDRESS
                        001A  5069 ;       LOOP COUNT I
                        001A  5070 ;
                        001A  5071 ;--
                        001A  5072 ;*******************************************************************************
                        001A  5073
                        001A  5074
                        001A  5075
                        001A  5076          INITIALIZE                              ;INIT THE CPU
                        001C  5077          LDFIELD         3$,,B,4$,15,15,         ;SET CLOCKX OFF
                        002D  5078          LOADDCS         4$,,03,1                ;LOAD INTO DCS
                        0034  5079          LOOP            I,1,4000                ;CREATE A TEST LOOP OF 4000
```

```
                        003B  5080          ERRLOOP                                 ;ERROR LOOP POINT
                        003D  5081          FETCH           0                       ;START DCS PROGRAM
                        0042  5082          BRSTCLK         4                       ;END DCS PROGRAM
                        0046  5083          CMPREGMSK       CSTRP,1$,,2$,,W,        ;TEST CS ADDRESS
                        0052  5084          IFERROR                                 ;INCORRECT BRANCH
                        0057  5085          ENDLOOP         I                       ;END TEST LOOP I
                        005A  5086          SKIP                                    ;GO TO NEXT TEST
                        0061  5087
                        0061  5088
                        0061  5089 BEGIN_TEST_DATA
                        0061
                        0061          ;-------------------------------------------------------------------------
                        0061  5090
                1800    0061  5091 1$:      .WORD   ^X1800                          ;EXPECTED RESULTS
                3FFF    0063  5092 2$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                  00    0065  5093 3$:      .BYTE   ^X00                            ;TURN OFF CLOCKX
                        0066  5094 4$:
U 03, 7701,0812,9C14,DED8,0470,1800     0066  5095 ;% 03   ROT/PL_MSS,WB_(M[MDR]+R[ZERO]).BCD,BUT/SRKSTA,NEXT/1800
                        0071  5099
                        0071  5100
                        0071  5101 BEGIN_CPU_DATA
                        0002  5102
                        0002  5103 DCS_DATA
                        0001  5104
U 00, 7700,C800,0364,0300,0470,1801     0001  5105 ;% 00   NOP
U 01, 7700,1800,C370,0300,C670,1802     000C  5109 ;% 01   LIT/LITRL,LITRL/1,ROT/ZLIT0,ALPCTL/WX_S,WCTRL/MDR_WB
U 02, 7700,5812,E110,03FF,C670,1803     0017  5113 ;% 02   MSRC/MDR,LIT/LITRL,LITRL/1FF,ROT/OLIT0,MUX/M.S,ALU/A+B+CI,WCTRL/MDR_WB
U 03, 7701,0812,9C14,DED8,0470,1800     0022  5117 ;% 03   ROT/PL_MSS,WB_(M[MDR]+R[ZERO]).BCD,BUT/SRKSTA,NEXT/1800
U 04, 7300,4800,0364,0300,0470,1805     002D  5121 ;% 04   NOP,STOP.CLOCK
                        0038  5125
                        0038  5126 RTEMP_DATA
                        0001  5127
            FFFFFFFF    0001  5128          .LONG   ^XFFFFFFFF
            FFFFFFFF    0005  5129          .LONG   ^XFFFFFFFF
            FFFFFFFF    0009  5130          .LONG   ^XFFFFFFFF
            FFFFFFFF    000D  5131          .LONG   ^XFFFFFFFF
            FFFFFFFF    0011  5132          .LONG   ^XFFFFFFFF
            FFFFFFFF    0015  5133          .LONG   ^XFFFFFFFF
            FFFFFFFF    0019  5134          .LONG   ^XFFFFFFFF
            FFFFFFFF    001D  5135          .LONG   ^XFFFFFFFF
                        0021  5136
                        0021  5137 END_CPU_DATA
                        0071  5138
                        0071  5139
                        0071  5140 END_TEST_DATA
                        0071
                        0071          ;-------------------------------------------------------------------------
                        0071  5141 ENDTEST
```

```
                        001C        .SBTTL  "TEST   0DE     FPA ENABLE/DISABLE TEST
                        001C  5143 ;*******************************************************************************
                        001C  5144 ;++
                        001C  5145 ;
                        001C  5146 ; FUNCTIONAL DESCRIPTION:
                        001C  5147 ;
                        001C  5148 ;       THIS TEST CHECKS THE FPA ENABLE/DISABLE FUNCTION
                        001C  5149 ;
                        001C  5150 ; TEST ALGORITHM:
                        001C  5151 ;
                        001C  5152 ;       1.  LOAD MICRO CODE INTO DCS
                        001C  5153 ;       2.  EXECUTE DCS PROGRAM
                        001C  5154 ;               a.  TEST FOR FPA PRESENT
                        001C  5155 ;       3.  IF FPA NOT PRESENT SKIP THE TEST
                        001C  5156 ;       4.  EXECUTE DCS PROGRAM
                        001C  5157 ;               a.  TEST FOR FPA ENABLED
                        001C  5158 ;       5.  IF FPA ENABLED LOAD NEW DCS MICRO CODE
                        001C  5159 ;       6.  EXECUTE DCS PROGRAM
                        001C  5160 ;               a.  DISABLE FPA
                        001C  5161 ;               b.  BRANCH ON FPA ENABLE
                        001C  5162 ;       7.  TEST CS ADDRESS FOR FPA DISABLED
                        001C  5163 ;       8.  IF NO ERROR MODIFY DCS MICRO CODE
                        001C  5164 ;       9.  EXECUTE DCS PROGRAM
                        001C  5165 ;               a.  ENABLE FPA
                        001C  5166 ;               b.  BRANCH ON FPA ENABLE
                        001C  5167 ;      10.  TEST CS ADDRESS FOR FPA ENABLED
                        001C  5168 ;      11.  IF NO ERROR MODIFY DCS MICRO CODE
                        001C  5169 ;      12.  EXECUTE DCS PROGRAM
                        001C  5170 ;               a.  DISABLE FPA
                        001C  5171 ;      13.  INIT THE FPA (PULL DC LO)
                        001C  5172 ;      14.  MODIFY DCS MICRO CODE
                        001C  5173 ;      15.  EXECUTE DCS PROGRAM
                        001C  5174 ;               a.  BRANCH ON FPA ENABLE
                        001C  5175 ;      16.  TEST CS ADDRESS FOR FPA ENABLED
                        001C  5176 ;      17.  IF NO ERROR GO TO NEXT TEST
                        001C  5177 ;
                        001C  5178 ;
                        001C  5179 ; TEST PATTERNS:
                        001C  5180 ;
                        001C  5181 ;       NONE
                        001C  5182 ;
                        001C  5183 ;
                        001C  5184 ; LOGIC DESCRIPTION:
                        001C  5185 ;
                        001C  5186 ;       THIS TEST FIRST CHECKS THAT DC LO (GENERATED BY INITIALIZE PSEUDO OP)
                        001C  5187 ;       CAN ENABLE THE FPA.  THEN THE FPA IS DISABLED AND ENABLED USING THE
                        001C  5188 ;       WCTRL MICRO ORDER.  IF REPLACING THE FPA DOES NOT FIX THE PROBLEM
                        001C  5189 ;       THEN THE DPM BRANCHING LOGIC SHOULD BE AT FAULT.
                        001C  5190 ;
                        001C  5191 ;
                        001C  5192 ; ASSUMPTIONS:
                        001C  5193 ;
                        001C  5194 ;       THIS TEST ASSUMES THE WBUS IS OPERATIONAL.
                        001C  5195 ;
                        001C  5196 ;
                        001C  5197 ; ERROR DESCRIPTION:
                        001C  5198 ;
```

```
                        001C  5199 ;       EXPECTED CS ADDRESS
                        001C  5200 ;       RECEIVED CS ADDRESS
                        001C  5201 ;
                        001C  5202 ;               NOTE: IF THE FORTH IFERROR STATEMENT FAILS THE MOST LIKELY
                        001C  5203 ;                     PROBLEM IS MSEQ INIT COMING FROM THE UBI MODULE.
                        001C  5204 ;--
                        001C  5205 ;*******************************************************************************
                        001C  5206
                        001C  5207
                        001C  5208          INITIALIZE                                  ;INIT THE CPU
                        001E  5209          LOADDCS         1$,,0,3                     ;LOAD MICRO CODE
                        0025  5210          FETCH           0                           ;EXECUTE DCS PROGRAM
                        002A  5211          BRSTCLK         2                           ;...
                        002E  5212          CMPREGMSK       CSTRP,2$,,2$,,W,NE          ;TEST FOR FPA PRESENT
                        003A  5213          SKIP            10$,NOERROR                 ;SKIP IF FPA PRESENT
                        0041  5214          PRINT_S         3$                          ;PRINT MESSAGE
                        0045  5215          SKIP                                        ;SKIP TEST NO FPA
                        004C  5216 10$:     ERRLOOP                                     ;ERROR LOOP POINT
                        004E  5217          INITIALIZE                                  ;PULL DC LOW
                        0050  5218          FETCH           0                           ;EXECUTE DCS PROGRAM
                        0055  5219          BRSTCLK         2                           ;...
                        0059  5220          CMPREGMSK       CSTRP,4$,,4$,,W,NE          ;TEST FOR FPA ENABLED
                        0065  5221          IFERROR         ,,<<FPA><DPM><MIC>>         ;FPA NOT ENABLED
                        006D  5222          LOADDCS         5$,,0,5                     ;LOAD MICRO CODE
                        0074  5223          ERRLOOP                                     ;ERROR LOOP POINT
                        0076  5224          FETCH           0                           ;EXECUTE DCS PROGRAM
                        007B  5225          BRSTCLK         4                           ;...
                        007F  5226          CMPREGMSK       CSTRP,4$,,4$,,W,            ;TEST FOR FPA DISABLED
                        008B  5227          IFERROR         ,,<<FPA><DPM><MIC>>         ;FPA NOT DISABLED
                        0093  5228          LOADDCS         6$,,02,1                    ;MODIFY MICRO CODE
                        009A  5229          ERRLOOP                                     ;ERROR LOOP POINT
                        009C  5230          FETCH           0                           ;EXECUTE DCS PROGRAM
                        00A1  5231          BRSTCLK         4                           ;...
                        00A5  5232          CMPREGMSK       CSTRP,4$,,4$,,W,NE          ;TEST FOR FPA ENABLED
                        00B1  5233          IFERROR         ,,<FPA>                     ;FPA NOT ENABLED
                        00B7  5234          LOADDCS         5$,,0,5                     ;LOAD MICRO CODE
                        00BE  5235          ERRLOOP                                     ;ERROR LOOP POINT
                        00C0  5236          FETCH           0                           ;EXECUTE DCS PROGRAM
                        00C5  5237          BRSTCLK         4,INH                       ;...
                        00C9  5238          INITIALIZE                                  ;INIT THE FPA
                        00CB  5239          LOADDCS         1$,,0,3                     ;LOAD DCS MICRO CODE
                        00D2  5240          FETCH           0                           ;EXECUTE DCS PROGRAM
                        00D7  5241          BRSTCLK         2,INH                       ;...
                        00DB  5242          CMPREGMSK       CSTRP,4$,,4$,,W,NE          ;TEST FOR FPA ENABLED
                        00E7  5243          IFERROR         ,,<<FPA><UBI>>              ;FPA NOT ENABLED
                        00EE  5244          SKIP                                        ;GO TO NEXT TEST
                        00F5  5245
                        00F5  5246
                        00F5  5247 BEGIN_TEST_DATA
                        00F5
                        00F5        ;-------------------------------------------------------------------------
                        00F5  5248
                        00F5  5249 1$:
U 00, 7700,C800,0364,0300,0470,1801  00F5  5250 ;% 00    NOP
U 01, 7700,4800,0364,C300,0470,1800  0100  5254 ;% 01    BUT/NO.FPA,NEXT/1800                ;TEST FPA STATUS
U 02, 7300,4800,0364,0300,0470,1803  010B  5258 ;% 02    NOP,STOP.CLOCK
                              0002   0116  5262 2$:      .WORD   ^X0002                      ;MASK FOR CMPREGMSK PSEUDO OP
```

```
ZZ-ECKAB-9.2    "TEST ODE FPA ENABLE/DISABLE TEST                    13-MAY-1987       Fiche 7 Frame H5          Sequence 1295
ECKAB                          VAX-11/750 DPM MICRO DIAGNOSTICS       18-MAR-1987 13:31:35  VAX/VMS Macro V04-00        Page 10
V09.02                         "TEST ODE FPA ENABLE/DISABLE TEST      18-MAR-1987 13:31:14  [VAX750.DPM]ECKAB6.MAR;1          (1

50 49 4B 53 20 54 53 45 54 0A 0D 00'  0118  5263 3$:     .ASCIC  <13><10>/TEST SKIPPED; FPA NOT PRESENT/<13><10>
54 4F 4E 20 41 50 46 20 3B 44 45 50   0124
      0A 0D 54 4E 45 53 45 52 50 20   0130
                                 21   0118
                               0001   013A  5264 4$:     .WORD   ^X0001                          ;MASK FOR CMPREGMSK PSEUDO OP
                                      013C  5265 5$:
U 00. 7700,C800,0364,0300,0470,1801   013C  5266 ;% 00   NOP
U 01. 7700,4800,0364,0300,0471,1802   0147  5270 ;% 01   FPA_M[TEMP0]                            ;TICK FPA OUT OF INIT STATE
U 02. 7700,4800,5BE4,03D8,2A70,1803   0152  5274 ;% 02   WCTRL/FPA.ENABLE_WB5,WB_R[ZERO]         ;DISABLE THE FPA
U 03. 7700,4800,0364,C300,0470,1800   015D  5278 ;% 03   BUT/NO.FPA,NEXT/1800                    ;TEST FPA STATUS
U 04. 7300,4800,0364,0300,0470,1805   0168  5282 ;% 04   NOP,STOP.CLOCK
                                      0173  5286 6$:
U 02. 7700,4800,5BE4,0300,2A70,1803   0173  5287 ;% 02   WCTRL/FPA.ENABLE_WB5,WB_R[TEMP0]         ;ENABLE THE FPA
                                      017E  5291
                                      017E  5292 BEGIN_CPU_DATA
                                      0002  5293
                                      0002  5294 RTEMP_DATA
                                      0001  5295
                           00000020   0001  5296          .LONG   ^X00000020                    ;ENABLE FPA
                                      0005  5297
                                      0005  5298 END_CPU_DATA
                                      017E  5299
                                      017E  5300 END_TEST_DATA
                                      017E
                                      017E        ;-------------------------------------------------------------------------
                                      017E  5301 ENDTEST
```

```
                        0026        .SBTTL  "TEST   0DF     TEST FPA STALL AND WAIT FUNCTIONS
                        0026  5303 ;*******************************************************************************
                        0026  5304 ;++
                        0026  5305 ;
                        0026  5306 ; FUNCTIONAL DESCRIPTION:
                        0026  5307 ;
                        0026  5308 ;       THIS TEST CHECKS THE FPA STALL SIGNAL COMING FROM THE FPA IN
                        0026  5309 ;       CONJUNCTION WITH THE FPA WAIT SIGNAL PRODUCED BY THE CPU MICROWORD.
                        0026  5310 ;
                        0026  5311 ;
                        0026  5312 ; TEST ALGORITHM:
                        0026  5313 ;
                        0026  5314 ;       1.  EXECUTE DCS PROGRAM
                        0026  5315 ;               a.  BRANCH ON FPA PRESENT
                        0026  5316 ;       2.  TEST CS ADDRESS FOR FPA PRESENT
                        0026  5317 ;       3.  IF FPA NOT PRESENT SKIP THIS TEST
                        0026  5318 ;       4.  EXECUTE DCS PROGRAM
                        0026  5319 ;               a.  DISABLE ^P INTERRUPTS
                        0026  5320 ;               b.  LOAD PC WITH ADDRESS ZERO TO CAUSE PREFETCH
                        0026  5321 ;               c.  DECODE THE FPA INSTRUCTION
                        0026  5322 ;               d.  ASSERT FPA WAIT
                        0026  5323 ;       5.  TEST CS ADDRESS FOR A STALL AT 1805
                        0026  5324 ;       6.  IF NO ERROR GO TO NEXT TEST
                        0026  5325 ;
                        0026  5326 ;
                        0026  5327 ; TEST PATTERNS:
                        0026  5328 ;
                        0026  5329 ;       NONE
                        0026  5330 ;
                        0026  5331 ;
                        0026  5332 ; LOGIC DESCRIPTION:
                        0026  5333 ;
                        0026  5334 ;       THIS TEST ENSURES THAT THE FPA STALL AND CPU WAIT SIGNALS ARE
                        0026  5335 ;       OPERATIONAL.  THE SAC GATE ARRAY ON DPM IS RESPONSIBLE FOR
                        0026  5336 ;       STALLING M CLOCK UNTIL THE FPA DEASSERTS STALL.
                        0026  5337 ;
                        0026  5338 ;
                        0026  5339 ; ASSUMPTIONS:
                        0026  5340 ;
                        0026  5341 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
                        0026  5342 ;
                        0026  5343 ;
                        0026  5344 ; ERROR DESCRIPTION:
                        0026  5345 ;
                        0026  5346 ;       EXPECTED CS ADDRESS
                        0026  5347 ;       RECEIVED CS ADDRESS
                        0026  5348 ;
                        0026  5349 ;--
                        0026  5350 ;*******************************************************************************
                        0026  5351
                        0026  5352
                        0026  5353
                        0026  5354          INITIALIZE                                  ;INIT THE CPU
                        0028  5355          FETCH           <^X07>                      ;START DCS PROGRAM
                        002D  5356          BRSTCLK         <^X09>                      ;END DCS PROGRAM
                        0031  5357          CMPREGMSK       CSTRP,1$,,1$,,W,NE          ;IS FPA PRESENT?
                        003D  5358          SKIP            10$,NOERROR                 ;SKIP IF PRESENT
```

```
                                0044  5359          PRINT_S         2$                      ;PRINT NO FPA
                                0048  5360          SKIP                                    ;GO TO NEXT TEST
                                004F  5361 10$:     EXPSTALL                                ;EXPECT M CLOCK TO STALL
                                0051  5362          ERRLOOP                                 ;ERROR LOOP POINT
                                0053  5363          FETCH           0                       ;START DCS PROGRAM
                                0058  5364          BRSTCLK         6                       ;END DCS PROGRAM
                                005C  5365          CMPREGMSK       CSTRP,3$,,4$,,W,        ;TEST FOR CORRECT CS ADDRESS
                                0068  5366          IFERROR         ,<SAC>,<<DPM><FPA>>     ;M CLOCK DID NOT STALL
                                0070  5367          SKIP                                    ;GO TO NEXT TEST
                                0077  5368
                                0077  5369
                                0077  5370 BEGIN_TEST_DATA
                                0077
                                0077          ;-----------------------------------------------------------------------
                                0077  5371
                          0002  0077  5372 1$:       .WORD   ^X0002                          ;FPA PRESENT
50 49 4B 53 20 54 53 45 54 0A 0D 00'  0079  5373 2$:       .ASCIC  <13><10>/TEST SKIPPED; FPA NOT PRESENT/<13><10>
54 4F 4E 20 41 50 46 20 3B 44 45 50   0085
      0A 0D 54 4E 45 53 45 52 50 20   0091
                               21     0079
                            1805  009B  5374 3$:       .WORD   ^X1805                          ;STALL ADDRESS
                            3FFF  009D  5375 4$:       .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                                009F  5376
                                009F  5377
                                009F  5378 BEGIN_CPU_DATA
                                0002  5379
                                0002  5380 DCS_DATA
                                0001  5381
U 00, 7700,C800,0364,0300,0470,1801   0001  5382 ;% 00    NOP
U 01, 7700,9800,D370,0360,2470,1802   000C  5386 ;% 01    CRAR_ZLIT16[0C0]                    ;DISABLE ^P INTERRUPTS
U 02, 7700,D800,D370,0320,2C70,1803   0017  5390 ;% 02    CONREGS_ZLIT16[040]                 ;...
U 03, 7700,4800,5BE4,03D8,4870,1804   0022  5394 ;% 03    PC_R[ZERO]                          ;CAUSE PREFETCH
U 04, 7700,4800,0364,1700,0470,1805   002D  5398 ;% 04    IRD1TEST                            ;CAUSE FPA TO DECODE INST.
U 05, 7700,6800,0364,0300,0474,1806   0038  5402 ;% 05    WB_FPA,FPAWAIT                      ;STALL CONDITION
U 06, 7300,C800,0364,0300,0470,1807   0043  5406 ;% 06    NOP,STOP.CLOCK                      ;STOP THE CPU CLOCK
U 07, 7700,C800,0364,0300,0470,1808   004E  5410 ;% 07    NOP
U 08, 7700,4800,0364,C300,0470,1800   0059  5414 ;% 08    BUT/NO.FPA,NEXT/1800                ;BRANCH ON FPA PRESENT
U 09, 7300,4800,0364,0300,0470,180A   0064  5418 ;% 09    NOP,STOP.CLOCK                      ;STOP THE CPU CLOCK
                                006F  5422
                                006F  5423 CACHE_DATA
                                0001  5424
                    00000051    0001  5425          .LONG   ^X00000051                      ;CMPF INSTRUCTION
                    00000000    0005  5426          .LONG   ^X00000000
                                0009  5427
                                0009  5428 END_CPU_DATA
                                009F  5429
                                009F  5430
                                009F  5431 END_TEST_DATA
                                009F
                                009F          ;-----------------------------------------------------------------------
                                009F  5432 ENDTEST
```

```
                        0021        .SBTTL  "TEST   0E0     FPA MBUS/WBUS INTERFACE TEST
                        0021  5434 ;*******************************************************************************
                        0021  5435 ;++
                        0021  5436 ;
                        0021  5437 ; FUNCTIONAL DESCRIPTION:
                        0021  5438 ;
                        0021  5439 ;       THIS TEST CHECKS THE DATA INTEGRITY OF THE MBUS AND WBUS INTERFACE
                        0021  5440 ;       TO THE FPA.
                        0021  5441 ;
                        0021  5442 ;
                        0021  5443 ; TEST ALGORITHM:
                        0021  5444 ;
                        0021  5445 ;       1.  EXECUTE DCS PROGRAM
                        0021  5446 ;               a.  BRANCH ON FPA PRESENT
                        0021  5447 ;       2.  TEST CS ADDRESS FOR FPA PRESENT
                        0021  5448 ;       3.  IF FPA NOT PRESENT SKIP TEST
                        0021  5449 ;       4.  CREATE A TEST LOOP OF 5
                        0021  5450 ;       5.  LOAD A TEST PATTERN INTO THE WDREG
                        0021  5451 ;       6.  EXECUTE DCS PROGRAM
                        0021  5452 ;               a.  MOVE TEST PATTERN TO RTEMP0
                        0021  5453 ;               b.  DISABLE ^P INTERRUPTS
                        0021  5454 ;               c.  LOAD PC WITH ADDRESS ZERO TO CAUSE PREFETCH
                        0021  5455 ;               d.  DECODE THE FPA INSTRUCTION
                        0021  5456 ;               e.  TICK THE FPA TO WAKE IT UP
                        0021  5457 ;               f.  PASS THE TEST PATTERN TO THE FPA ON W AND M BUSSES
                        0021  5458 ;               g.  WAIT FOR THE FPA TO COMPLETE THE INSTRUCTION
                        0021  5459 ;               h.  PASS THE RESULTS TO THE RDM
                        0021  5460 ;       7.  TEST THE RESULTS
                        0021  5461 ;       8.  IF NO ERROR GO TO STEP 5 UNTIL THE TEST LOOP IS EXHAUSTED
                        0021  5462 ;       9.  LOAD A TEST PATTERN INTO THE WDREG
                        0021  5463 ;      10.  EXECUTE MICRO CODE IN STEP 6
                        0021  5464 ;      11.  TEST THE RESULTS
                        0021  5465 ;      12.  IF NO ERROR GO TO NEXT TEST
                        0021  5466 ;
                        0021  5467 ;
                        0021  5468 ; TEST PATTERNS:
                        0021  5469 ;
                        0021  5470 ;       AAAAAAAA
                        0021  5471 ;       55555555
                        0021  5472 ;       33333333
                        0021  5473 ;       0F0F0F0F
                        0021  5474 ;       00FF00FF
                        0021  5475 ;       0000FEFF
                        0021  5476 ;
                        0021  5477 ;
                        0021  5478 ; LOGIC DESCRIPTION:
                        0021  5479 ;
                        0021  5480 ;       THIS TEST CHECKS THE DATA INTEGRITY OF THE M AND W BUSSES TO THE
                        0021  5481 ;       FPA MODULE.  TEST PATTERNS ARE PASSED TO THE FPA WHERE AN ADDF2
                        0021  5482 ;       IS PERFORMED.  THE RESULTS PASSED BACK ARE CHECKED FOR THE CORRECT
                        0021  5483 ;       PATTERN.  ERRORS SHOULD BE CONFINED TO THE FPA, DPM, MIC OR THE
                        0021  5484 ;       BACKPLANE.
                        0021  5485 ;
                        0021  5486 ;
                        0021  5487 ; ASSUMPTIONS:
                        0021  5488 ;
                        0021  5489 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
```

```
                                                0021  5490 ;
                                                0021  5491 ;
                                                0021  5492 ; ERROR DESCRIPTION:
                                                0021  5493 ;
                                                0021  5494 ;       EXPECTED DATA FROM FPA
                                                0021  5495 ;       RECEIVED DATA FROM FPA
                                                0021  5496 ;       LOOP COUNT I
                                                0021  5497 ;
                                                0021  5498 ;--
                                                0021  5499 ;*****************************************************************************
                                                0021  5500
                                                0021  5501
                                                0021  5502          INITIALIZE                              ;INIT THE CPU
                                                0023  5503          FETCH           <^X0A>                  ;START DCS PROGRAM
                                                0028  5504          BRSTCLK         <^X0C>                  ;END DCS PROGRAM
                                                002C  5505          CMPREGMSK       CSTRP,1$,,1$,,W,NE      ;IS FPA PRESENT?
                                                0038  5506          SKIP            10$,NOERROR             ;SKIP IF PRESENT
                                                003F  5507          PRINT_S         2$                      ;PRINT NO FPA
                                                0043  5508          SKIP                                    ;GO TO NEXT TEST
                                                004A  5509 10$:     LOOP            I,1,5                   ;TEST LOOP OF 6
                                                0051  5510          SA01PAT         I,3$                    ;CREATE A TEST PATTERN
                                                0059  5511          LOADREG         WDREG,3$,,L             ;PUT PATTERN IN WDREG
                                                0061  5512          ERRLOOP                                 ;ERROR LOOP POINT
                                                0063  5513          FETCH           0                       ;START DCS PROGRAM
                                                0068  5514          BRSTCLK         9                       ;END DCS PROGRAM
                                                006C  5515          CMPREG          WHREG,4$,I,L,           ;TEST FPA DATA
                                                0075  5516          IFERROR         ,,<<FPA><DPM><MIC>>     ;INCORRECT DATA
                                                007D  5517          ENDLOOP         I                       ;END TEST LOOP I
                                                0080  5518          LOADREG         WDREG,5$,,L             ;LOAD TEST PATTERN INTO WDREG
                                                0088  5519          ERRLOOP                                 ;ERROR LOOP POINT
                                                008A  5520          FETCH           0                       ;START DCS PROGRAM
                                                008F  5521          BRSTCLK         9                       ;END DCS PROGRAM
                                                0093  5522          CMPREG          WHREG,6$,,L,            ;TEST FPA DATA
                                                009C  5523          IFERROR         ,,<<FPA><DPM><MIC>>     ;INCORRECT DATA
                                                00A4  5524          SKIP                                    ;GO TO NEXT TEST
                                                00AB  5525
                                                00AB  5526
                                                00AB  5527 BEGIN_TEST_DATA
                                                00AB
                                                00AB       ;-------------------------------------------------------------------------
                                                00AB  5528
                                      0002      00AB  5529 1$:      .WORD   ^X0002                          ;FPA PRESENT BIT
50 49 4B 53 20 54 53 45 54 0A 0D 00'            00AD  5530 2$:      .ASCIC  <13><10>/TEST SKIPPED; FPA NOT PRESENT/<13><10>
54 4F 4E 20 41 50 46 20 3B 44 45 50             00B9
       0A 0D 54 4E 45 53 45 52 50 20            00C5
                                   21           00AD
                                00000000        00CF  5531 3$:      .LONG   ^X00000000                      ;TEST PATTERN STORAGE
                                AAAAAB2A        00D3  5532 4$:      .LONG   ^XAAAAAB2A                      ;RESULTS TABLE
                                555555D5        00D7  5533          .LONG   ^X555555D5
                                333333B3        00DB  5534          .LONG   ^X333333B3
                                0F0F0F8F        00DF  5535          .LONG   ^X0F0F0F8F
                                00FF017F        00E3  5536          .LONG   ^X00FF017F
                                0000FEFF        00E7  5537 5$:      .LONG   ^X0000FEFF                      ;LAST TEST PATTERN
                                0000FF7F        00EB  5538 6$:      .LONG   ^X0000FF7F                      ;LAST RESULTS
                                                00EF  5539
                                                00EF  5540
                                                00EF  5541 BEGIN_CPU_DATA
```

```
                                0002  5542
                                0002  5543 DCS_DATA
                                0001  5544
U 00, 7700,C800,0364,0300,0470,1801   0001  5545 ;% 00   NOP
U 01, 5700,8840,0364,0300,0470,1802   000C  5549 ;% 01   R[TEMP0]_RDM                          ;TEST PATTERN TO R & M TEMP0
U 02, 7700,1800,D370,0360,2470,1803   0017  5553 ;% 02   CRAR_ZLI16[0C0]                      ;DISABLE ^P INTERRUPTS
U 03, 7700,5800,D370,0320,2C70,1804   0022  5557 ;% 03   CONREGS_ZLI16[040]                  ;...
U 04, 7700,C800,5BE4,03D8,4870,1805   002D  5561 ;% 04   PC_R[ZERO]                          ;PREFETCH INSTRUCTION
U 05, 7700,4800,0364,1700,0470,1806   0038  5565 ;% 05   IRD1TEST                            ;CAUSE FPA TO DECODE INST.
U 06, 7700,4800,0364,0300,0471,1807   0043  5569 ;% 06   FPA_M[TEMP0]                        ;TICK FPA OUT OF INIT STATE
U 07, 7700,C800,5BE4,0300,0473,1808   004E  5573 ;% 07   FPA_M[TEMP0] FPA_R[TEMP0]          ;PASS DATA TO FPA
U 08, 6700,6800,0364,0300,0474,1809   0059  5577 ;% 08   RDM_WB,WB_FPA,FPAWAIT               ;PASS RESULTS TO RDM
U 09, 7300,4800,0364,0300,0470,180A   0064  5581 ;% 09   NOP,STOP.CLOCK                      ;STOP CPU CLOCK
U 0A, 7700,C800,0364,0300,0470,180B   006F  5585 ;% 0A   NOP
U 0B, 7700,4800,0364,C300,0470,1800   007A  5589 ;% 0B   BUT/NO.FPA,NEXT/1800               ;TEST FPA STATUS
U 0C, 7300,C800,0364,0300,0470,180D   0085  5593 ;% 0C   NOP,STOP.CLOCK                      ;STOP CPU CLOCK
                                0090  5597
                                0090  5598
                                0090  5599 CACHE_DATA
                                0001  5600
                     00000040   0001  5601         .LONG   ^X00000040                     ;ADDF2 INSTRUCTION
                     00000000   0005  5602         .LONG   ^X00000000
                                0009  5603
                                0009  5604 END_CPU_DATA
                                00EF  5605
                                00EF  5606 END_TEST_DATA
                                00EF
                                00EF        ;-----------------------------------------------------------------------------
                                00EF  5607 ENDTEST
```

```
0023        .SBTTL  "TEST   0E1     TEST FPA RESERVED OPERAND TRAP
0023  5609 ;*******************************************************************************
0023  5610 ;++
0023  5611 ;
0023  5612 ; FUNCTIONAL DESCRIPTION:
0023  5613 ;
0023  5614 ;       THIS TEST CHECKS THE OPERATION OF THE FPA RESERVED OPERAND MICROTRAP.
0023  5615 ;
0023  5616 ;
0023  5617 ; TEST ALGORITHM:
0023  5618 ;
0023  5619 ;       1.  EXECUTE DCS PROGRAM
0023  5620 ;               a.  BRANCH ON FPA PRESENT
0023  5621 ;       2.  IF FPA NOT PRESENT SKIP TEST
0023  5622 ;       3.  EXECUTE DCS PROGRAM
0023  5623 ;               a.  DISABLE ^P INTERRUPTS
0023  5624 ;               b.  LOAD PC WITH ADDRESS ZERO TO CAUSE PREFETCH
0023  5625 ;               c.  DECODE FPA INSTRUCTION
0023  5626 ;               d.  TICK THE FPA OUT OF INIT STATE
0023  5627 ;               e.  PASS DATA TO FPA OVER M AND W BUSSES
0023  5628 ;               f.  PASS RESULTS BACK TO THE RDM
0023  5629 ;       4.  CHECK THE MICRO VECTOR LINES FOR THE CORRECT BRANCH
0023  5630 ;       5.  IF NO ERROR TEST CS ADDRESS FOR CORRECT TRAP ADDRESS
0023  5631 ;       6.  IF NO ERROR EXECUTE DCS PROGRAM
0023  5632 ;               a.  READ FLOATING POINT TRAP CODE REGISTER TO RDM
0023  5633 ;       7.  TEST TRAP CODE REGISTER CONTENTS
0023  5634 ;       8.  IF NO ERROR GO TO NEXT TEST
0023  5635 ;
0023  5636 ; TEST PATTERNS:
0023  5637 ;
0023  5638 ;       NONE
0023  5639 ;
0023  5640 ;
0023  5641 ; LOGIC DESCRIPTION:
0023  5642 ;
0023  5643 ;       THIS TEST ENSURES THAT THE FPA RESERVED OPERAND TRAP SIGNAL IS
0023  5644 ;       TRANSMITTED BY THE FPA TO THE MIC MODULE.  THIS CAUSES LOGIC
0023  5645 ;       ON MIC AND DPM TO CREATE A MICROTRAP TO CS ADDRESS 2C.
0023  5646 ;
0023  5647 ;
0023  5648 ; ASSUMPTIONS:
0023  5649 ;
0023  5650 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
0023  5651 ;
0023  5652 ;
0023  5653 ; ERROR DESCRIPTION:
0023  5654 ;
0023  5655 ;       FIRST IFERROR:
0023  5656 ;               EXPECTED VBUS DATA
0023  5657 ;               RECEIVED VBUS DATA
0023  5658 ;                       (BITS <7:0> = VBUS BIT POSITION)
0023  5659 ;                       (BITS <8> = BIT STATE)
0023  5660 ;
0023  5661 ;       SECOND IFERROR:
0023  5662 ;               EXPECTED CS ADDRESS
0023  5663 ;               RECEIVED CS ADDRESS
0023  5664 ;
```

```
                                                0023  5665 ;       THIRD IFERROR:
                                                0023  5666 ;               EXPECTED FPA TRAP CODE REGISTER CONTENTS
                                                0023  5667 ;               RECEIVED FPA TRAP CODE REGISTER CONTENTS
                                                0023  5668 ;
                                                0023  5669 ;--
                                                0023  5670 ;*****************************************************************************
                                                0023  5671
                                                0023  5672
                                                0023  5673         INITIALIZE                                      ;INIT THE CPU
                                                0025  5674         FETCH           <^X0C>                          ;START DCS PROGRAM
                                                002A  5675         BRSTCLK         <^X0E>                          ;END DCS PROGRAM
                                                002E  5676         CMPREGMSK       CSTRP,1$,,1$,,W,NE              ;IS FPA PRESENT?
                                                003A  5677         SKIP            10$,NOERROR                     ;SKIP IF PRESENT
                                                0041  5678         PRINT_S         2$                              ;PRINT NO FPA
                                                0045  5679         SKIP                                            ;GO TO NEXT TEST
                                                004C  5680 10$:     EXPUTRAP                                        ;EXPECT A MICROTRAP
                                                004E  5681         ERRLOOP                                         ;ERROR LOOP POINT
                                                0050  5682         FETCH           0                               ;START DCS PROGRAM
                                                0055  5683         BRSTCLK         8                               ;END DCS PROGRAM
                                                0059  5684         CMPVBUS         3$                              ;TEST MICRO VECTOR LINES
                                                005E  5685         IFERROR         ,<<ACV><UTR>>,<<FPA><MIC>>
                                                0067  5686         CMPREGMSK       CSTRP,4$,,5$,,W,                ;TEST CS ADDRESS
                                                0073  5687         IFERROR         ,,<DPM>                         ;INCORRECT TRAP ADDRESS
                                                0079  5688         FETCH           <^X09>                          ;START DCS PROGRAM
                                                007E  5689         BRSTCLK         <^X0B>                          ;END DCS PROGRAM
                                                0082  5690         CMPREGMSK       WHREG,6$,,7$,,L,                ;TEST TRAP CODE
                                                008E  5691         IFERROR         ,,<FPA>                         ;WRONG CODE
                                                0094  5692         SKIP
                                                009B  5693
                                                009B  5694
                                                009B  5695 BEGIN_TEST_DATA
                                                009B
                                                009B        ;-------------------------------------------------------------------------
                                                009B  5696
                                          0002  009B  5697 1$:       .WORD   ^X0002                                  ;FPA PRESENT BIT
50 49 4B 53 20 54 53 45 54 0A 0D 00'      009D  5698 2$:       .ASCIC  <13><10>/TEST SKIPPED; FPA NOT PRESENT/<13><10>
54 4F 4E 20 41 50 46 20 3B 44 45 50       00A9
      0A 0D 54 4E 45 53 45 52 50 20       00B5
                                    21    009D
                                    04    00BF  5699 3$:       .BYTE   ^X04
                                          00C0  5700           VBUSDATA        <^X08>,1                        ;MICRO VECTOR LINES
                                          00C1  5701           VBUSDATA        <^X09>,1
                                          00C2  5702           VBUSDATA        <^X0A>,0
                                          00C3  5703           VBUSDATA        <^X0B>,0
                                    002C  00C4  5704 4$:       .WORD   ^X002C                                  ;TRAP ADDRESS
                                    3FFF  00C6  5705 5$:       .WORD   ^X3FFF                                  ;MASK FOR CMPREGMSK PSEUDO OP
                              00000003  00C8  5706 6$:       .LONG   ^X00000003                              ;TRAP CODE REGISTER CONTENTS
                              00000007  00CC  5707 7$:       .LONG   ^X00000007                              ;MASK FOR CMPREGMSK PSEUDO OP
                                          00D0  5708
                                          00D0  5709
                                          00D0  5710 BEGIN_CPU_DATA
                                          0002  5711
                                          0002  5712 DCS_DATA
                                          0001  5713
U 00, 7700,C800,0364,0300,0470,1801       0001  5714 ;% 00    NOP
U 01, 7700,9800,D370,0360,2470,1802       000C  5718 ;% 01    CRAR_ZLIT16[0C0]                                ;DISABLE ^P INTERRUPTS
U 02, 7700,D800,D370,0320,2C70,1803       0017  5722 ;% 02    CONREGS_ZLIT16[040]                             ;...
```

```
U 03. 7700.4800.5BE4.03D8.4870.1804  0022  5726 ;% 03   PC_R[ZERO]                                   ;CAUSE PREFETCH
U 04. 7700.4800.0364.1700.0470.1805  002D  5730 ;% 04   IRD1TEST                                    ;START INSTRUCTION DECODE
U 05. 7700.C800.0364.0300.0471.1806  0038  5734 ;% 05   FPA_M[TEMP0]                                ;TICK FPA OUT OF INIT STATE
U 06. 7700.C800.5BE4.0300.0473.1807  0043  5738 ;% 06   FPA_M[TEMP0] FPA_R[TEMP0]                   ;PASS DATA TO RDM
U 07. 6600.E800.0364.0300.0474.1808  004E  5742 ;% 07   RDM_WB,WB_FPA,FPAWAIT,STROBE.V.BUS         ;READ RESULTS TO RDM
U 08. 7300.4800.0364.0300.0470.1809  0059  5746 ;% 08   NOP,STOP.CLOCK                               ;STOP THE CPU CLOCK
U 09. 7700.4800.0364.0300.0470.180A  0064  5750 ;% 09   NOP
U 0A. 6700.4800.0364.0300.0670.180B  006F  5754 ;% 0A   RDM_WB,WCTRL/FPTCR                           ;READ TRAP CODE REGISTER
U 0B. 7300.4800.0364.0300.0470.180C  007A  5758 ;% 0B   NOP,STOP.CLOCK                               ;STOP THE CPU CLOCK
U 0C. 7700.C800.0364.0300.0470.180D  0085  5762 ;% 0C   NOP
U 0D. 7700.4800.0364.C300.0470.1800  0090  5766 ;% 0D   BUT/NO.FPA,NEXT/1800                        ;TEST FPA STATUS
U 0E. 7300.4800.0364.0300.0470.180F  009B  5770 ;% 0E   NOP,STOP.CLOCK                               ;STOP THE CPU CLOCK
                                     00A6  5774
                                     00A6  5775 RTEMP_DATA
                                     0001  5776
                         FFFFFFFF    0001  5777          .LONG   ^XFFFFFFFF                       ;FPA DATA
                                     0005  5778
                                     0005  5779 CACHE_DATA
                                     0001  5780
                         00000040    0001  5781          .LONG   ^X00000040                       ;ADDF2 INSTRUCTION
                         00000000    0005  5782          .LONG   ^X00000000
                                     0009  5783
                                     0009  5784 END_CPU_DATA
                                     00D0  5785
                                     00D0  5786
                                     00D0  5787 END_TEST_DATA
                                     00D0
                                     00D0          ;-----------------------------------------------------------------------------
                                     00D0  5788 ENDTEST
```

```
                        0018        .SBTTL  "TEST   0E2     TEST FPA TRAP LOGIC
                        0018  5790 ;*******************************************************************************
                        0018  5791 ;++
                        0018  5792 ;
                        0018  5793 ; FUNCTIONAL DESCRIPTION:
                        0018  5794 ;
                        0018  5795 ;       THIS TEST VERIFIES THAT THE FPA TRAP SIGNAL CAUSES AN INTERRUPT
                        0018  5796 ;       AT THE CPU.
                        0018  5797 ;
                        0018  5798 ;
                        0018  5799 ; TEST ALGORITHM:
                        0018  5800 ;
                        0018  5801 ;       1.  EXECUTE DCS PROGRAM
                        0018  5802 ;               a.  BRANCH ON FPA PRESENT
                        0018  5803 ;       2.  TEST FOR FPA PRESENT
                        0018  5804 ;       3.  IF FPA NOT PRESENT SKIP THIS TEST
                        0018  5805 ;       4.  EXECUTE DCS PROGRAM
                        0018  5806 ;               a.  DISABLE ^P INTERRUPTS
                        0018  5807 ;               b.  LOAD PC WITH ZERO TO CAUSE PREFETCH
                        0018  5808 ;               c.  TICK THE FPA OUT OF IT'S INIT STATE
                        0018  5809 ;               d.  LOAD THE FPA IV BIT
                        0018  5810 ;               e.  DECODE THE FPA INSTRUCTION
                        0018  5811 ;               f.  TICK THE FPA OUT OF INIT STATE
                        0018  5812 ;               g.  PASS DATA TO THE FPA
                        0018  5813 ;               h.  WAIT FOR FPA TO PROCESS THE INSTRUCTION
                        0018  5814 ;               i.  ALLOW INTERRUPT TO HAPPEN
                        0018  5815 ;               j.  STROBE THE VBUS FOR THE INTERRUPT
                        0018  5816 ;       5.  TEST THE VBUS FOR DO SRVC L
                        0018  5817 ;       6.  IF NO ERROR TEST THE CS ADDRESS FOR AN FPA ARITHMETIC TRAP
                        0018  5818 ;       7.  IF NO ERROR TEST THE FPA ANSWER
                        0018  5819 ;       8.  IF NO ERROR GO TO NEXT TEST
                        0018  5820 ;
                        0018  5821 ;
                        0018  5822 ; TEST PATTERNS:
                        0018  5823 ;
                        0018  5824 ;       NONE
                        0018  5825 ;
                        0018  5826 ;
                        0018  5827 ; LOGIC DESCRIPTION:
                        0018  5828 ;
                        0018  5829 ;       THIS TEST CHECKS TO SEE IF THE FPA CAN GENERATE AN FPA ARITHMETIC
                        0018  5830 ;       TRAP AND THAT THE CPU RESPONDS CORRECTLY TO IT.
                        0018  5831 ;
                        0018  5832 ;
                        0018  5833 ; ASSUMPTIONS:
                        0018  5834 ;
                        0018  5835 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
                        0018  5836 ;
                        0018  5837 ;
                        0018  5838 ; ERROR DESCRIPTION:
                        0018  5839 ;
                        0018  5840 ;       FIRST IFERROR:
                        0018  5841 ;               EXPECTED VBUS DATA
                        0018  5842 ;               RECEIVED VBUS DATA
                        0018  5843 ;                       (BITS <7:0> = VBUS BIT POSITION)
                        0018  5844 ;                       (BITS <8> = VBUS BIT STATE)
                        0018  5845 ;
```

```
                                0018  5846 ;       SECOND IFERROR:
                                0018  5847 ;              EXPECTED CS ADDRESS
                                0018  5848 ;              RECEIVED CS ADDRESS
                                0018  5849 ;
                                0018  5850 ;       THIRD IFERROR:
                                0018  5851 ;              EXPECTED FPA RESULTS
                                0018  5852 ;              RECEIVED FPA RESULTS
                                0018  5853 ;
                                0018  5854 ;--
                                0018  5855 ;*****************************************************************************
                                0018  5856
                                0018  5857
                                0018  5858         INITIALIZE                                      ;INIT THE CPU
                                001A  5859         FETCH           <^X0D>                          ;START DCS PROGRAM
                                001F  5860         BRSTCLK         <^X0F>                          ;END DCS PROGRAM
                                0023  5861         CMPREGMSK       CSTRP,1$,,1$,,W,NE              ;IS FPA PRESENT?
                                002F  5862         SKIP            10$,NOERROR                     ;SKIP IF PRESENT
                                0036  5863         PRINT_S         2$                              ;PRINT NO FPA
                                003A  5864         SKIP                                            ;GO TO NEXT TEST
                                0041  5865 10$:     EXPUTRAP                                        ;EXPECT A TRAP
                                0043  5866         ERRLOOP                                         ;ERROR LOOP POINT
                                0045  5867         FETCH           0                               ;START DCS PROGRAM
                                004A  5868         BRSTCLK         <^X0C>                          ;END DCS PROGRAM
                                004E  5869         CMPVBUS         3$                              ;TEST FOR DO SERVICE
                                0053  5870         IFERROR         ,<SAC>,<<FPA><DPM><MIC>>;SIGNAL MISSING
                                005C  5871         CMPREGMSK       CSTRP,4$,,5$,,W,                ;TEST FOR TRAP ADDRESS
                                0068  5872         IFERROR         ,<SAC>,<DPM>                    ;INCORRECT ADDRESS
                                006F  5873         CMPREG          WHREG,6$,,L,                    ;TEST FPA ANSWER
                                0078  5874         IFERROR         ,,<FPA>                         ;INCORRECT RESULTS
                                007E  5875         SKIP                                            ;GO TO NEXT TEST
                                0085  5876
                                0085  5877
                                0085  5878 BEGIN_TEST_DATA
                                0085
                                0085       ;-----------------------------------------------------------------------
                                0085  5879
                          0002  0085  5880 1$:      .WORD   ^X0002                                  ;FPA PRESENT
50 49 4B 53 20 54 53 45 54 0A 0D 00'  0087  5881 2$:      .ASCIC  <13><10>/TEST SKIPPED; FPA NOT PRESENT/<13><10>
54 4F 4E 20 41 50 46 20 3B 44 45 50   0093
      0A 0D 54 4E 45 53 45 52 50 20   009F
                               21     0087
                               01     00A9  5882 3$:      .BYTE   ^X01
                                      00AA  5883          VBUSDATA        <^X19>,0                ;DO SRVC L
                             0012     00AB  5884 4$:      .WORD   ^X0012                          ;FPA ARITHMETIC TRAP ADD.
                             3FFF     00AD  5885 5$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                         00000100     00AF  5886 6$:      .LONG   ^X00000100                      ;FPA ANSWER
                                      00B3  5887
                                      00B3  5888 BEGIN_CPU_DATA
                                      0002  5889
                                      0002  5890 DCS_DATA
                                      0001  5891
U 00, 7700,C800,0364,0300,0470,1801   0001  5892 ;* 00    NOP
U 01, 7700,9800,D370,0360,2470,1802   000C  5896 ;* 01    CRAR_ZLIT16[0C0]                        ;DISABLE ^P INTERRUPTS
U 02, 7700,D800,D370,0320,2C70,1803   0017  5900 ;* 02    CONREGS_ZLIT16[040]                     ;...
U 03, 7700,4800,5BE4,03D8,4870,1804   0022  5904 ;* 03    PC_R[ZERO]                              ;CAUSE PREFETCH
U 04, 7700,8801,0364,0300,0471,1805   002D  5908 ;* 04    FPA_M[TEMP1]                            ;TICK FPA OUT OF INIT STATE
U 05, 7700,8800,5BF4,0300,0070,1806   0038  5912 ;* 05    PSL_R[TEMP0]                            ;LOAD PSL IV BIT
```

```
U 06. 7700.C800.0364.1700.0470.1807  0043  5916 ;% 06   IRD1TEST                                  ;LOAD INST. INTO FPA
U 07. 7700.0801.0364.0300.0471.1808  004E  5920 ;% 07   FPA_M[TEMP1]                              ;TICK FPA OUT OF INIT STATE
U 08. 7700.8801.0364.0300.0471.1809  0059  5924 ;% 08   FPA_M[TEMP1]                              ;PASS DATA TO FPA
U 09. 6700.6800.0364.0300.0474.180A  0064  5928 ;% 09   RDM_FPA,FPAWAIT                          ;READ ANSWER BACK
U 0A. 7700.C800.0364.1700.0470.180B  006F  5932 ;% 0A   IRD1TEST
U 0B. 7600.4800.0364.0300.0470.180C  007A  5936 ;% 0B   NOP,STROBE.V.BUS
U 0C. 7300.C800.0364.0300.0470.180D  0085  5940 ;% 0C   NOP,STOP.CLOCK                            ;STOP THE CPU CLOCK
U 0D. 7700.C800.0364.0300.0470.180E  0090  5944 ;% 0D   NOP
U 0E. 7700.4800.0364.C300.0470.1800  009B  5948 ;% 0E   BUT/NO.FPA,NEXT/1800                     ;TEST FPA STATE
U 0F. 7300.C800.0364.0300.0470.1810  00A6  5952 ;% 0F   NOP,STOP.CLOCK                            ;STOP THE CPU CLOCK
                                     00B1  5956
                                     00B1  5957 RTEMP_DATA
                                     0001  5958
                         00000020    0001  5959          .LONG   ^X00000020                    ;IV BIT FOR PSL
                         00004480    0005  5960          .LONG   ^X00004480                    ;DATA FOR FPA
                                     0009  5961
                                     0009  5962 CACHE_DATA
                                     0001  5963
                         00000048    0001  5964          .LONG   ^X00000048                    ;CVTFB INSTRUCTION
                         00000000    0005  5965          .LONG   ^X00000000
                                     0009  5966
                                     0009  5967 END_CPU_DATA
                                     00B3  5968
                                     00B3  5969
                                     00B3  5970 END_TEST_DATA
                                     00B3
                                     00B3       ;-------------------------------------------------------------------------
                                     00B3  5971 ENDTEST
```

```
001F        .SBTTL  "TEST   0E3     FPA CONDITION CODES TEST 1
001F  5973 ;*******************************************************************************
001F  5974 ;++
001F  5975 ;
001F  5976 ; FUNCTIONAL DESCRIPTION:
001F  5977 ;
001F  5978 ;       THIS TEST CHECKS THE FPA CONDITION CODE LOGIC IN THE CCC GATE ARRAY.
001F  5979 ;
001F  5980 ;
001F  5981 ; TEST ALGORITHM:
001F  5982 ;
001F  5983 ;       1.  EXECUTE DCS PROGRAM
001F  5984 ;               a.  BRANCH ON FPA PRESENT
001F  5985 ;       2.  TEST FOR FPA PRESENT
001F  5986 ;       3.  IF FPA NOT PRESENT SKIP TEST
001F  5987 ;       4.  CREATE A TEST LOOP OF 2
001F  5988 ;       5.  MODIFY DCS MICROCODE ACCORDING TO LOOP COUNT
001F  5989 ;       6.  EXECUTE DCS PROGRAM
001F  5990 ;               a.  DISABLE ^P INTERRUPTS
001F  5991 ;               b.  ZERO THE PSL REGISTER
001F  5992 ;               c.  ZERO THE PC REGISTER TO CAUSE PREFETCH
001F  5993 ;               d.  DECODE THE FPA INSTRUCTION
001F  5994 ;               e.  TICK THE FPA TO WAKE IT UP
001F  5995 ;               f.  PASS DATA TO THE FPA ON M AND W BUSSES
001F  5996 ;               g.  WAIT FOR FPA TO FINISH AND SET CONDITION CODES
001F  5997 ;               h.  READ PSL TO THE RDM
001F  5998 ;       7.  TEST CONDITION OF Z AND V BITS
001F  5999 ;       8.  IF NO ERROR GO TO STEP 5 FOR LAST LOOP
001F  6000 ;       9.  GO TO NEXT TEST
001F  6001 ;
001F  6002 ;
001F  6003 ; TEST PATTERNS:
001F  6004 ;
001F  6005 ;       ITERATION          MBUS            WBUS
001F  6006 ;           1           00000000        00000000
001F  6007 ;           2           00000000        00004000
001F  6008 ;
001F  6009 ;
001F  6010 ; LOGIC DESCRIPTION:
001F  6011 ;
001F  6012 ;       THIS TEST CHECKS THE OPERATION OF THE Z BIT USING THE COMPARE FLOATING
001F  6013 ;       INSTRUCTION.  THE CCC GATE ARRAY SHOULD SET THIS BIT ACCORDING TO
001F  6014 ;       THE FPA'S COPY OF THE Z BIT (FPA Z L).
001F  6015 ;
001F  6016 ;
001F  6017 ; ASSUMPTIONS:
001F  6018 ;
001F  6019 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
001F  6020 ;
001F  6021 ;
001F  6022 ; ERROR DESCRIPTION:
001F  6023 ;
001F  6024 ;       EXPECTED PSL DATA
001F  6025 ;       RECEIVED PSL DATA
001F  6026 ;       LOOP COUNT
001F  6027 ;
001F  6028 ;--
```

```
                                  001F  ;*****************************************************************************
                                  001F
                                  001F
                                  001F          INITIALIZE                                  ;INIT THE CPU
                                  0021          FETCH           <^X0B>                      ;START DCS PROGRAM
                                  0026          BRSTCLK         <^X0D>                      ;END DCS PROGRAM
                                  002A          CMPREGMSK       CSTRP,1$,,1$,,W,NE          ;IS FPA PRESENT?
                                  0036          SKIP            10$,NOERROR                 ;SKIP IF PRESENT
                                  003D          PRINT_S         2$                          ;PRINT NO FPA
                                  0041          SKIP                                        ;GO TO NEXT TEST
                                  0048  10$:    LOOP            I,1,2                       ;TEST LOOP OF 2
                                  004F          LOADDCS         3$,I,07,1                   ;MODIFY DCS PROGRAM
                                  0056          ERRLOOP                                     ;ERROR LOOP POINT
                                  0058          FETCH           0                           ;START DCS PROGRAM
                                  005D          BRSTCLK         <^X0A>                      ;END DCS PROGRAM
                                  0061          CMPREGMSK       WHREG,4$,I,5$,,L,           ;TEST PSL Z AND V BITS
                                  006D          IFERROR         ,<CCC>,<<FPA><DPM><MIC>>;Z BIT INCORRECT
                                  0076          ENDLOOP         I                           ;END TEST LOOP I
                                  0079          SKIP                                        ;GO TO NEXT TEST
                                  0080
                                  0080
                                  0080  BEGIN_TEST_DATA
                                  0080
                                  0080  ;-----------------------------------------------------------------------
                                  0080
                            0002  0080  1$:     .WORD   ^X0002                              ;FPA PRESENT
50 49 4B 53 20 54 53 45 54 0A 0D 00'  0082  2$:     .ASCIC  <13><10>/TEST SKIPPED; FPA NOT PRESENT/<13><10>
54 4F 4E 20 41 50 46 20 3B 44 45 50   008E
      0A 0D 54 4E 45 53 45 52 50 20   009A
                                 21   0082
                                  00A4  3$:
U 07, 7700,C800,5BE4,0300,0473,1808   00A4  ;% 07   FPA_M[TEMP0] FPA_R[TEMP0]                  ;I = 1
U 07, 7700,8800,5BE4,0304,0473,1808   00AF  ;% 07   FPA_M[TEMP0] FPA_R[TEMP1]                  ;I = 2
                         00000004   00BA  4$:     .LONG   ^X00000004                          ;RESULTS TABLE
                         00000000   00BE          .LONG   ^X00000000                          ;...
                         00000006   00C2  5$:     .LONG   ^X00000006                          ;MASK FOR CMPREGMSK PSEUDO OP
                                  00C6
                                  00C6  BEGIN_CPU_DATA
                                  0002
                                  0002  DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801   0001  ;% 00   NOP
U 01, 7700,9800,D370,0360,2470,1802   000C  ;% 01   CRAR_ZLIT16[0C0]                            ;DISABLE ^P INTERRUPTS
U 02, 7700,D800,D370,0320,2C70,1803   0017  ;% 02   CONREGS_ZLIT16[040]                         ;...
U 03, 7700,0800,5BF4,03D8,0070,1804   0022  ;% 03   PSL_R[ZERO]                                 ;ZERO THE PSL
U 04, 7700,C800,5BE4,03D8,4870,1805   002D  ;% 04   PC_R[ZERO]                                  ;PREFETCH INSTRUCTION
U 05, 7700,4800,0364,1700,0470,1806   0038  ;% 05   IRD1TEST                                    ;CAUSE FPA TO DECODE INST.
U 06, 7700,4800,0364,0300,0471,1807   0043  ;% 06   FPA_M[TEMP0]                                ;TICK FPA OUT OF INIT STATE
U 07, 7700,C800,5BE4,0300,0473,1808   004E  ;% 07   FPA_M[TEMP0] FPA_R[TEMP0]                  ;PASS DATA TO FPA
U 08, 7701,2800,0364,0301,0474,1809   0059  ;% 08   WB_FPA,FPAWAIT,CCOP2,SIZE[IDEP]             ;SET THE CONDITION CODES
U 09, 6700,4800,0364,0300,0870,180A   0064  ;% 09   RDM_WB,WB_PSL                               ;READ PSL TO RDM
U 0A, 7300,C800,0364,0300,0470,180B   006F  ;% 0A   NOP,STOP.CLOCK                              ;STOP CPU CLOCK
U 0B, 7700,4800,0364,0300,0470,180C   007A  ;% 0B   NOP
U 0C, 7700,4800,0364,C300,0470,1800   0085  ;% 0C   BUT/NO.FPA,NEXT/1800                        ;TEST FPA STATUS
U 0D, 7300,C800,0364,0300,0470,180E   0090  ;% 0D   NOP,STOP.CLOCK                              ;STOP CPU CLOCK
                                  009B
                                  009B
```

```
                    009B  6129 RTEMP_DATA
                    0001  6130
00000000            0001  6131          .LONG   ^X00000000                      ;DATA FOR FPA
00004000            0005  6132          .LONG   ^X00004000                      ;...
                    0009  6133
                    0009  6134 CACHE_DATA
                    0001  6135
00000051            0001  6136          .LONG   ^X00000051                      ;CMPF INSTRUCTION
00000000            0005  6137          .LONG   ^X00000000
                    0009  6138
                    0009  6139 END_CPU_DATA
                    00C6  6140
                    00C6  6141 END_TEST_DATA
                    00C6
                    00C6          ;-----------------------------------------------------------------------
                    00C6  6142 ENDTEST
```

```
                        001F        .SBTTL  "TEST   0E4     FPA CONDITION CODES TEST 2
                        001F  6144 ;*******************************************************************************
                        001F  6145 ;++
                        001F  6146 ;
                        001F  6147 ; FUNCTIONAL DESCRIPTION:
                        001F  6148 ;
                        001F  6149 ;       THIS TEST CHECKS THE CONDITION CODE LOGIC IN THE CCC GATE ARRAY.
                        001F  6150 ;
                        001F  6151 ;
                        001F  6152 ; TEST ALGORITHM:
                        001F  6153 ;
                        001F  6154 ;       1.  EXECUTE DCS PROGRAM
                        001F  6155 ;               a.  BRANCH ON FPA PRESENT
                        001F  6156 ;       2.  TEST FOR FPA PRESENT
                        001F  6157 ;       3.  IF FPA NOT PRESENT SKIP TEST
                        001F  6158 ;       4.  EXPECT A MICRO TRAP CONDITION
                        001F  6159 ;       5.  CREATE A TEST LOOP OF 2
                        001F  6160 ;       6.  MODIFY DCS MICROCODE ACCORDING TO LOOP COUNT
                        001F  6161 ;       7.  EXECUTE DCS PROGRAM
                        001F  6162 ;               a.  DISABLE ^P INTERRUPTS
                        001F  6163 ;               b.  ZERO THE PSL REGISTER
                        001F  6164 ;               c.  LOAD THE PC WITH ZERO TO CAUSE PREFETCH
                        001F  6165 ;               d.  TICK THE FPA TO WAKE IT UP
                        001F  6166 ;               e.  PASS DATA TO THE FPA VIA M AND W BUSSES
                        001F  6167 ;               f.  WAIT FOR FPA TO FINISH AND SET CONDITION CODES
                        001F  6168 ;               g.  READ PSL TO THE RDM
                        001F  6169 ;       8.  TEST PSL REGISTER CONTENTS
                        001F  6170 ;       9.  IF NO ERROR GO TO STEP 6 FOR LAST LOOP
                        001F  6171 ;      10.  GO TO NEXT TEST
                        001F  6172 ;
                        001F  6173 ;
                        001F  6174 ; TEST PATTERNS:
                        001F  6175 ;
                        001F  6176 ;       ITERATION          MBUS            WBUS
                        001F  6177 ;           1           00004080        0000C080
                        001F  6178 ;           2           00007F80        00007F80
                        001F  6179 ;
                        001F  6180 ;
                        001F  6181 ; LOGIC DESCRIPTION:
                        001F  6182 ;
                        001F  6183 ;       THIS TEST CHECKS THE SETTING AND CLEARING OF THE Z AND V BITS IN
                        001F  6184 ;       THE CCC GATE ARRAY BY THE FPA.  THE ADDF2 INSTRUCTION IS USED TO
                        001F  6185 ;       SETUP THE PROPER CONDITIONS.
                        001F  6186 ;
                        001F  6187 ;
                        001F  6188 ; ASSUMPTIONS:
                        001F  6189 ;
                        001F  6190 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
                        001F  6191 ;
                        001F  6192 ;
                        001F  6193 ; ERROR DESCRIPTION:
                        001F  6194 ;
                        001F  6195 ;       EXPECTED PSL DATA
                        001F  6196 ;       RECEIVED PSL DATA
                        001F  6197 ;       LOOP COUNT
                        001F  6198 ;
                        001F  6199 ;--
```

```
                        001F  6200 ;*****************************************************************************
                        001F  6201
                        001F  6202
                        001F  6203          INITIALIZE                                  ;INIT THE CPU
                        0021  6204          FETCH           <^X0C>                      ;START DCS PROGRAM
                        0026  6205          BRSTCLK         <^X0E>                      ;END DCS PROGRAM
                        002A  6206          CMPREGMSK       CSTRP,1$,,1$,,W,NE          ;IS FPA PRESENT?
                        0036  6207          SKIP            10$,NOERROR                 ;SKIP IF PRESENT
                        003D  6208          PRINT_S         2$                          ;PRINT NO FPA
                        0041  6209          SKIP                                        ;GO TO NEXT TEST
                        0048  6210 10$:     EXPUTRAP                                    ;EXPECT A MICRO TRAP
                        004A  6211          LOOP            I,1,2                       ;TEST LOOP OF 2
                        0051  6212          LOADDCS         3$,I,07,1                   ;MODIFY DCS PROGRAM
                        0058  6213          LOADDCS         6$,I,09,1                   ;...
                        005F  6214          ERRLOOP                                     ;ERROR LOOP POINT
                        0061  6215          FETCH           0                           ;START DCS PROGRAM
                        0066  6216          BRSTCLK         <^X0B>                      ;END DCS PROGRAM
                        006A  6217          FETCH           <^X09>                      ;START DCS PROGRAM
                        006F  6218          BRSTCLK         <^X0B>                      ;END DCS PROGRAM
                        0073  6219          CMPREGMSK       WHREG,4$,I,5$,,L,           ;TEST PSL Z AND V BITS
                        007F  6220          IFERROR         ,<CCC>,<<FPA><DPM>>         ;Z OR V BIT INCORRECT
                        0087  6221          ENDLOOP         I                           ;END TEST LOOP I
                        008A  6222          SKIP                                        ;GO TO NEXT TEST
                        0091  6223
                        0091  6224
                        0091  6225 BEGIN_TEST_DATA
                        0091
                        0091              ;-----------------------------------------------------------------------
                        0091  6226
                   0002 0091  6227 1$:      .WORD   ^X0002                              ;FPA PRESENT
50 49 4B 53 20 54 53 45 54 0A 0D 00' 0093  6228 2$:      .ASCIC  <13><10>/TEST SKIPPED; FPA NOT PRESENT/<13><10>
54 4F 4E 20 41 50 46 20 3B 44 45 50  009F
      0A 0D 54 4E 45 53 45 52 50 20  00AB
                                21    0093
                        00B5  6229 3$:
U 07, 7700,C801,5BE4,0308,0473,1808 00B5  6230 ;% 07   FPA_M[TEMP1] FPA_R[TEMP2]          ;I = 1
U 07, 7700,C800,5BE4,0300,0473,1808 00C0  6234 ;% 07   FPA_M[TEMP0] FPA_R[TEMP0]          ;I = 2
                   00000004 00CB  6238 4$:      .LONG   ^X00000004                      ;RESULTS TABLE
                   00000002 00CF  6239          .LONG   ^X00000002                      ;...
                   00000006 00D3  6240 5$:      .LONG   ^X00000006                      ;MASK FOR CMPREGMSK PSEUDO OP
                        00D7  6241 6$:
U 09, 7700,4800,0364,0300,0470,180A 00D7  6242 ;% 09   NOP                             ;I = 1
U 09, 7701,0800,0364,0301,0470,180A 00E2  6246 ;% 09   CCOP2,SIZE[IDEP]                 ;I = 2
                        00ED  6250
                        00ED  6251 BEGIN_CPU_DATA
                        0002  6252
                        0002  6253 DCS_DATA
                        0001  6254
U 00, 7700,C800,0364,0300,0470,1801 0001  6255 ;% 00   NOP
U 01, 7700,9800,D370,0360,2470,1802 000C  6259 ;% 01   CRAR_ZLIT16[0C0]                 ;DISABLE ^P INTERRUPTS
U 02, 7700,D800,D370,0320,2C70,1803 0017  6263 ;% 02   CONREGS_ZLIT16[040]              ;...
U 03, 7700,0800,5BF4,03D8,0070,1804 0022  6267 ;% 03   PSL_R[ZERO]                      ;ZERO THE PSL
U 04, 7700,C800,5BE4,03D8,4870,1805 002D  6271 ;% 04   PC_R[ZERO]                       ;PREFETCH INSTRUCTION
U 05, 7700,4800,0364,1700,0470,1806 0038  6275 ;% 05   IRD1TEST                         ;CAUSE FPA TO DECODE INST.
U 06, 7700,4800,0364,0300,0471,1807 0043  6279 ;% 06   FPA_M[TEMP0]                    ;TICK FPA OUT OF INIT STATE
U 07, 7700,C800,5BE4,0300,0473,1808 004E  6283 ;% 07   FPA_M[TEMP0] FPA_R[TEMP0]       ;PASS DATA TO FPA
U 08, 7701,2800,0364,0301,0474,1809 0059  6287 ;% 08   WB_FPA,FPAWAIT,CCOP2,SIZE[IDEP]  ;SET THE CONDITION CODES
```

```
U 09, 7700,4800,0364,0300,0470,180A  0064  6291 ;% 09   NOP                                    ;...
U 0A, 6700,C800,0364,0300,0870,180B  006F  6295 ;% 0A   RDM_WB,WB_PSL                          ;READ PSL TO RDM
U 0B, 7300,4800,0364,0300,0470,180C  007A  6299 ;% 0B   NOP,STOP.CLOCK                        ;STOP CPU CLOCK
U 0C, 7700,C800,0364,0300,0470,180D  0085  6303 ;% 0C   NOP
U 0D, 7700,4800,0364,C300,0470,1800  0090  6307 ;% 0D   BUT/NO.FPA,NEXT/1800                 ;TEST FPA STATUS
U 0E, 7300,4800,0364,0300,0470,180F  009B  6311 ;% 0E   NOP,STOP.CLOCK                        ;STOP CPU CLOCK
                                     00A6  6315
                                     00A6  6316
                                     00A6  6317 RTEMP_DATA
                                     0001  6318
                           00007F80  0001  6319         .LONG   ^X00007F80                   ;DATA FOR FPA
                           00004080  0005  6320         .LONG   ^X00004080                   ;...
                           0000C080  0009  6321         .LONG   ^X0000C080                   ;...
                                     000D  6322
                                     000D  6323 CACHE_DATA
                                     0001  6324
                           00000040  0001  6325         .LONG   ^X00000040                   ;ADDF2 INSTRUCTION
                           00000000  0005  6326         .LONG   ^X00000000
                                     0009  6327
                                     0009  6328 END_CPU_DATA
                                     00ED  6329
                                     00ED  6330 END_TEST_DATA
                                     00ED
                                     00ED       ;-------------------------------------------------------------------------
                                     00ED  6331 ENDTEST
```

```
001F        .SBTTL  "TEST   0E5     FPA CONDITION CODES TEST 3
001F   6333 ;*******************************************************************************
001F   6334 ;++
001F   6335 ;
001F   6336 ; FUNCTIONAL DESCRIPTION:
001F   6337 ;
001F   6338 ;       THIS TEST CHECKS THE FPA CONDITION CODE LOGIC IN THE CCC GATE ARRAY.
001F   6339 ;
001F   6340 ;
001F   6341 ; TEST ALGORITHM:
001F   6342 ;
001F   6343 ;       1.  EXECUTE DCS PROGRAM
001F   6344 ;               a.  BRANCH ON FPA PRESENT
001F   6345 ;       2.  TEST FOR FPA PRESENT
001F   6346 ;       3.  IF FPA NOT PRESENT SKIP TEST
001F   6347 ;       4.  CREATE A TEST LOOP OF 2
001F   6348 ;       5.  MODIFY DCS MICROCODE ACCORDING TO LOOP COUNT
001F   6349 ;       6.  EXECUTE DCS PROGRAM
001F   6350 ;               a.  DISABLE ^P INTERRUPTS
001F   6351 ;               b.  ZERO THE PSL REGISTER
001F   6352 ;               c.  LOAD PC WITH ZERO TO CAUSE PREFETCH
001F   6353 ;               d.  DECODE THE FPA INSTRUCTION
001F   6354 ;               e.  PASS DATA TO THE FPA VIA THE MBUS
001F   6355 ;               f.  WAIT FOR FPA TO FINISH AND SET CONDITION CODES
001F   6356 ;               g.  READ THE PSL TO THE RDM
001F   6357 ;       7.  TEST THE PSL FOR THE CORRECT CONDITION CODES
001F   6358 ;       8.  IF NO ERROR GO TO STEP 5 FOR LAST LOOP
001F   6359 ;       9.  GO TO NEXT TEST
001F   6360 ;
001F   6361 ;
001F   6362 ; TEST PATTERNS:
001F   6363 ;
001F   6364 ;       ITERATION       MBUS DATA
001F   6365 ;           1           00000000
001F   6366 ;                       00000000
001F   6367 ;                       00000000
001F   6368 ;
001F   6369 ;           2           00004000
001F   6370 ;                       00004000
001F   6371 ;                       00004000
001F   6372 ;
001F   6373 ;
001F   6374 ; LOGIC DESCRIPTION:
001F   6375 ;
001F   6376 ;       THIS TEST CHECKS THAT THE CCC GATE ARRAY DECODES THE EMODF INSTRUCTION
001F   6377 ;       CORRECTLY.  THE CONDITION CODES ARE THEN SET AS A RESULT OF THE FPA
001F   6378 ;       Z L SIGNAL.
001F   6379 ;
001F   6380 ;
001F   6381 ; ASSUMPTIONS:
001F   6382 ;
001F   6383 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
001F   6384 ;
001F   6385 ;
001F   6386 ; ERROR DESCRIPTION:
001F   6387 ;
001F   6388 ;       EXPECTED PSL DATA
```

```
                        001F  6389 ;       RECEIVED PSL DATA
                        001F  6390 ;       LOOP COUNT
                        001F  6391 ;
                        001F  6392 ;--
                        001F  6393 ;*******************************************************************************
                        001F  6394
                        001F  6395
                        001F  6396         INITIALIZE                                      ;INIT THE CPU
                        0021  6397         FETCH           <^X0E>                          ;START DCS PROGRAM
                        0026  6398         BRSTCLK         <^X10>                          ;END DCS PROGRAM
                        002A  6399         CMPREGMSK       CSTRP,1$,,1$,,W,NE              ;IS FPA PRESENT?
                        0036  6400         SKIP            10$,NOERROR                     ;SKIP IF PRESENT
                        003D  6401         PRINT_S         2$                              ;PRINT NO FPA
                        0041  6402         SKIP                                            ;GO TO NEXT TEST
                        0048  6403 10$:     LOOP            I,1,2                           ;TEST LOOP OF 2
                        004F  6404         LOADDCS         3$,I,07,3                       ;MODIFY DCS PROGRAM
                        0056  6405         ERRLOOP                                         ;ERROR LOOP POINT
                        0058  6406         FETCH           0                               ;START DCS PROGRAM
                        005D  6407         BRSTCLK         <^X0D>                          ;END DCS PROGRAM
                        0061  6408         CMPREGMSK       WHREG,4$,I,5$,,L,               ;TEST PSL Z AND V BITS
                        006D  6409         IFERROR         ,<CCC>,<<FPA><DPM><MIC>>;Z BIT INCORRECT
                        0076  6410         ENDLOOP         I                               ;END TEST LOOP I
                        0079  6411         SKIP                                            ;GO TO NEXT TEST
                        0080  6412
                        0080  6413
                        0080  6414 BEGIN_TEST_DATA
                        0080
                        0080        ;-------------------------------------------------------------------------
                        0080  6415
                  0002  0080  6416 1$:      .WORD   ^X0002                                  ;FPA PRESENT
50 49 4B 53 20 54 53 45 54 0A 0D 00'  0082  6417 2$:      .ASCIC  <13><10>/TEST SKIPPED; FPA NOT PRESENT/<13><10>
54 4F 4E 20 41 50 46 20 3B 44 45 50   008E
      0A 0D 54 4E 45 53 45 52 50 20   009A
                              21      0082
                        00A4  6418 3$:
U 07, 7700,4800,0364,0300,0471,1808   00A4  6419 ;% 07   FPA_M[TEMP0]                      ;I = 1
U 08, 7700,C800,0364,0300,0471,1809   00AF  6423 ;% 08   FPA_M[TEMP0]                      ;...
U 09, 7700,C800,0364,0300,0471,180A   00BA  6427 ;% 09   FPA_M[TEMP0]                      ;...
U 07, 7700,0801,0364,0300,0471,1808   00C5  6431 ;% 07   FPA_M[TEMP1]                      ;I = 2
U 08, 7700,8801,0364,0300,0471,1809   00D0  6435 ;% 08   FPA_M[TEMP1]                      ;...
U 09, 7700,8801,0364,0300,0471,180A   00DB  6439 ;% 09   FPA_M[TEMP1]                      ;...
                        00000004      00E6  6443 4$:      .LONG   ^X00000004              ;RESULTS TABLE
                        00000000      00EA  6444         .LONG   ^X00000000              ;...
                        00000006      00EE  6445 5$:      .LONG   ^X00000006              ;MASK FOR CMPREGMSK PSEUDO OP
                        00F2  6446
                        00F2  6447 BEGIN_CPU_DATA
                        0002  6448
                        0002  6449 DCS_DATA
                        0001  6450
U 00, 7700,C800,0364,0300,0470,1801   0001  6451 ;% 00   NOP
U 01, 7700,9800,D370,0360,2470,1802   000C  6455 ;% 01   CRAR_ZLIT16[0C0]                 ;DISABLE ^P INTERRUPTS
U 02, 7700,D800,D370,0320,2C70,1803   0017  6459 ;% 02   CONREGS_ZLIT16[040]             ;...
U 03, 7700,0800,5BF4,03D8,0070,1804   0022  6463 ;% 03   PSL_R[ZERO]                     ;ZERO THE PSL
U 04, 7700,C800,5BE4,03D8,4870,1805   002D  6467 ;% 04   PC_R[ZERO]                      ;PREFETCH INSTRUCTION
U 05, 7700,4800,0364,1700,0470,1806   0038  6471 ;% 05   IRD1TEST                        ;CAUSE FPA TO DECODE INST.
U 06, 7700,4800,0364,0300,0471,1807   0043  6475 ;% 06   FPA_M[TEMP0]                    ;TICK FPA OUT OF INIT STATE
U 07, 7700,4800,0364,0300,0471,1808   004E  6479 ;% 07   FPA_M[TEMP0]                    ;PASS DATA TO FPA
```

```
U 08, 7700,C800,0364,0300,0471,1809  0059  6483 ;% 08   FPA_M[TEMP0]                                  ;...
U 09, 7700,C800,0364,0300,0471,180A  0064  6487 ;% 09   FPA_M[TEMP0]                                  ;...
U 0A, 7700,E800,0364,0300,0474,180B  006F  6491 ;% 0A   WB_FPA,FPAWAIT                               ;WAIT FOR FPA TO FINISH
U 0B, 7701,2800,0364,0301,0474,180C  007A  6495 ;% 0B   WB_FPA,FPAWAIT,CCOP2,SIZE[IDEP]             ;SET THE CONDITION CODES
U 0C, 6700,C800,0364,0300,0870,180D  0085  6499 ;% 0C   RDM_WB,WB_PSL                                ;READ PSL TO RDM
U 0D, 7300,C800,0364,0300,0470,180E  0090  6503 ;% 0D   NOP,STOP.CLOCK                               ;STOP CPU CLOCK
U 0E, 7700,4800,0364,0300,0470,180F  009B  6507 ;% 0E   NOP
U 0F, 7700,4800,0364,C300,0470,1800  00A6  6511 ;% 0F   BUT/NO.FPA,NEXT/1800                         ;TEST FPA STATUS
U 10, 7300,4800,0364,0300,0470,1811  00B1  6515 ;% 10   NOP,STOP.CLOCK                               ;STOP CPU CLOCK
                                     00BC  6519
                                     00BC  6520
                                     00BC  6521 RTEMP_DATA
                                     0001  6522
                         00000000    0001  6523          .LONG   ^X00000000                       ;DATA FOR FPA
                         00004000    0005  6524          .LONG   ^X00004000                       ;...
                                     0009  6525
                                     0009  6526 CACHE_DATA
                                     0001  6527
                         00000054    0001  6528          .LONG   ^X00000054                       ;EMODF INSTRUCTION
                         00000000    0005  6529          .LONG   ^X00000000
                                     0009  6530
                                     0009  6531 END_CPU_DATA
                                     00F2  6532
                                     00F2  6533 END_TEST_DATA
                                     00F2
                                     00F2        ;-------------------------------------------------------------------------
                                     00F2  6534 ENDTEST
```

```
                        .SBTTL  "TEST   0E6     FPA CONDITION CODES TEST 4
001F   6536 ;*******************************************************************************
001F   6537 ;++
001F   6538 ;
001F   6539 ; FUNCTIONAL DESCRIPTION:
001F   6540 ;
001F   6541 ;       THIS TEST CHECKS THE CONDITION CODE LOGIC IN THE CCC GATE ARRAY.
001F   6542 ;
001F   6543 ;
001F   6544 ; TEST ALGORITHM:
001F   6545 ;
001F   6546 ;       1.  EXECUTE DCS PROGRAM
001F   6547 ;               a.  BRANCH ON FPA PRESENT
001F   6548 ;       2.  TEST FOR FPA PRESENT
001F   6549 ;       3.  IF FPA NOT PRESENT SKIP TEST
001F   6550 ;       4.  CREATE A TEST LOOP OF 2
001F   6551 ;       5.  MODIFY DCS MICROCODE ACCORDING TO LOOP COUNT
001F   6552 ;       6.  MOVE LOOP COUNT INTO WDREG
001F   6553 ;       7.  EXECUTE DCS PROGRAM
001F   6554 ;               a.  DISABLE ^P INTERRUPTS
001F   6555 ;               b.  SET IPL TO 1F AND NATIVE MODE
001F   6556 ;               c.  FORM A CACHE ADDRESS FROM THE LOOP COUNT
001F   6557 ;               d.  LOAD PC WITH ADDRESS TO FORCE PREFETCH
001F   6558 ;               e.  DECODE FPA INSTRUCTION
001F   6559 ;               f.  TRY TO SET CONDITION CODES
001F   6560 ;               g.  READ PSL TO RDM
001F   6561 ;       8.  TEST PSL REGISTER CONTENTS
001F   6562 ;       9.  IF NO ERROR GO TO STEP 5 FOR LAST LOOP
001F   6563 ;      10.  GO TO NEXT TEST
001F   6564 ;
001F   6565 ;
001F   6566 ; TEST PATTERNS:
001F   6567 ;
001F   6568 ;       ITERATION       SIZE    WBUS DATA
001F   6569 ;           1           WORD    FFFF0000
001F   6570 ;           2           BYTE    FFFFFF00
001F   6571 ;
001F   6572 ;
001F   6573 ; LOGIC DESCRIPTION:
001F   6574 ;
001F   6575 ;       THIS TEST CHECKS THAT THE Z BIT CANNOT BE SET BY THE CPU WHEN A
001F   6576 ;       FPA INSTRUCTION IS DECODED AND THE FPA IS PRESENT AND ENABLED.
001F   6577 ;
001F   6578 ;
001F   6579 ; ASSUMPTIONS:
001F   6580 ;
001F   6581 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
001F   6582 ;
001F   6583 ;
001F   6584 ; ERROR DESCRIPTION:
001F   6585 ;
001F   6586 ;       EXPECTED PSL DATA
001F   6587 ;       RECEIVED PSL DATA
001F   6588 ;       LOOP COUNT
001F   6589 ;
001F   6590 ;--
001F   6591 ;*******************************************************************************
```

```
                                001F  6592
                                001F  6593
                                001F  6594          INITIALIZE                                  ;INIT THE CPU
                                0021  6595          FETCH           <^X0D>                      ;START DCS PROGRAM
                                0026  6596          BRSTCLK         <^X0F>                      ;END DCS PROGRAM
                                002A  6597          CMPREGMSK       CSTRP,1$,,1$,,W,NE          ;IS FPA PRESENT?
                                0036  6598          SKIP            10$,NOERROR                 ;SKIP IF PRESENT
                                003D  6599          PRINT_S         2$                          ;PRINT NO FPA
                                0041  6600          SKIP                                        ;GO TO NEXT TEST
                                0048  6601 10$:      LOOP            I,1,2                       ;TEST LOOP OF 2
                                004F  6602          LOADDCS         3$,I,<^X0A>,1               ;MODIFY DCS MICRO CODE
                                0056  6603          SAVEINDEX       I,6$                        ;SAVE LOOP COUNT
                                005B  6604          LOADREG         WDREG,6$,,L                 ;MOVE COUNT TO WDREG
                                0063  6605          ERRLOOP                                     ;ERROR LOOP POINT
                                0065  6606          FETCH           0                           ;START DCS PROGRAM
                                006A  6607          BRSTCLK         <^X0C>                      ;END DCS PROGRAM
                                006E  6608          CMPREGMSK       WHREG,4$,,5$,,L,            ;TEST PSL Z BIT
                                007A  6609          IFERROR         ,<CCC>,<<FPA><DPM><MIC>>;Z BIT INCORRECT
                                0083  6610          ENDLOOP         I                           ;END TEST LOOP
                                0086  6611          SKIP                                        ;GO TO NEXT TEST
                                008D  6612
                                008D  6613
                                008D  6614 BEGIN_TEST_DATA
                                008D
                                008D        ;-------------------------------------------------------------------------
                                008D  6615
                                0002  008D  6616 1$:      .WORD   ^X0002                          ;FPA PRESENT
50 49 4B 53 20 54 53 45 54 0A 0D 00'  008F  6617 2$:      .ASCIC  <13><10>/TEST SKIPPED; FPA NOT PRESENT/<13><10>
54 4F 4E 20 41 50 46 20 3B 44 45 50   009B
      0A 0D 54 4E 45 53 45 52 50 20   00A7
                                21    008F
                                00B1  6618 3$:
U 0A, 7701,8800,5BE4,0109,0470,180B   00B1  6619 ;% 0A   SIZE[WORD],WB_R[TEMP2],CCOP2             ;I = 1
U 0A, 7701,8800,5BE4,000D,0470,180B   00BC  6623 ;% 0A   SIZE[BYTE],WB_R[TEMP3],CCOP2             ;I = 2
                          00000000    00C7  6627 4$:     .LONG   ^X00000000                      ;Z BIT STATE
                          00000004    00CB  6628 5$:     .LONG   ^X00000004                      ;MASK FOR CMPREGMSK PSEUDO OP
                          00000000    00CF  6629 6$:     .LONG   ^X00000000                      ;LOOP COUNT STORAGE
                                00D3  6630
                                00D3  6631
                                00D3  6632 BEGIN_CPU_DATA
                                0002  6633
                                0002  6634 DCS_DATA
                                0001  6635
U 00, 7700,C800,0364,0300,0470,1801   0001  6636 ;% 00   NOP
U 01, 7700,9800,D370,0360,2470,1802   000C  6640 ;% 01   CRAR_ZLIT16[0C0]                         ;DISABLE ^P INTERRUPTS
U 02, 7700,D800,D370,0320,2C70,1803   0017  6644 ;% 02   CONREGS_ZLIT16[040]                     ;...
U 03, 7700,0800,5BF4,0300,0070,1804   0022  6648 ;% 03   PSL_R[TEMP0]                            ;SET IPL TO 1F & NATIVE MODE
U 04, 5700,4840,0364,0304,0470,1805   002D  6652 ;% 04   R[TEMP1]_RDM                            ;PASS LOOP COUNT TO RTEMP1
U 05, 7700,D800,EF64,0301,0470,1806   0038  6656 ;% 05   PL_[2]                                 ;LOAD 2 INTO PLATCH
U 06, 7700,C800,8B71,0304,0470,1807   0043  6660 ;% 06   Q_R[TEMP1].RL.P                         ;FORM CACHE ADDRESS FROM LOOP
U 07, 7700,4800,03A4,03D8,4870,1808   004E  6664 ;% 07   WCTRL/PC_WB,WB_Q                        ;CAUSE PREFETCH
U 08, 7700,4800,0364,1700,0470,1809   0059  6668 ;% 08   IRD1TEST                               ;LOAD INSTRUCTION INTO IR
U 09, 7700,4800,0364,0300,0470,180A   0064  6672 ;% 09   NOP                                     ;...
U 0A, 7701,8800,5BE4,0109,0470,180B   006F  6676 ;% 0A   SIZE[WORD],WB_R[TEMP2],CCOP2             ;SET CONDITION CODES
U 0B, 6700,4800,0364,0300,0870,180C   007A  6680 ;% 0B   RDM_WB,WB_PSL                           ;READ PSL TO RDM
U 0C, 7300,C800,0364,0300,0470,180D   0085  6684 ;% 0C   NOP,STOP.CLOCK                          ;STOP THE CPU CLOCK
U 0D, 7700,C800,0364,0300,0470,180E   0090  6688 ;% 0D   NOP
```

```
U 0E, 7700,4800,0364,C300,0470,1800  009B  6692 ;% 0E   BUT/NO.FPA,NEXT/1800                    ;BRANCH ON FPA PRESENT
U 0F, 7300,C800,0364,0300,0470,1810  00A6  6696 ;% 0F   NOP,STOP.CLOCK                          ;STOP THE CPU CLOCK
                                     00B1  6700
                                     00B1  6701 RTEMP_DATA
                                     0001  6702
                           001F0000  0001  6703         .LONG   ^X001F0000                     ;DATA FOR PSL
                           00000000  0005  6704         .LONG   ^X00000000                     ;LOOP COUNT STORAGE
                           FFFF0000  0009  6705         .LONG   ^XFFFF0000                     ;TEST DATA
                           FFFFFF00  000D  6706         .LONG   ^XFFFFFF00                     ;...
                                     0011  6707
                                     0011  6708 CACHE_DATA
                                     0001  6709
                           0000004E  0001  6710         .LONG   ^X0000004E                     ;CVTLF INSTRUCTION
                           00000040  0005  6711         .LONG   ^X00000040                     ;ADDF2 INSTRUCTION
                                     0009  6712
                                     0009  6713 END_CPU_DATA
                                     00D3  6714
                                     00D3  6715
                                     00D3  6716 END_TEST_DATA
                                     00D3
                                     00D3          ;-----------------------------------------------------------------------
                                     00D3  6717 ENDTEST
```

```
0025        .SBTTL  "TEST   0E7     TEST FPA COPY OF CONDITION CODES
0025  6719 ;*******************************************************************************
0025  6720 ;++
0025  6721 ;
0025  6722 ; FUNCTIONAL DESCRIPTION:
0025  6723 ;
0025  6724 ;       THIS TEST CHECKS THE COPY OF THE CONDITION CODES KEPT IN THE FPA MODULE
0025  6725 ;
0025  6726 ;
0025  6727 ; TEST ALGORITHM:
0025  6728 ;
0025  6729 ;       1.  EXECUTE DCS PROGRAM
0025  6730 ;               a.  BRANCH ON FPA PRESENT
0025  6731 ;       2.  TEST FOR FPA PRESENT
0025  6732 ;       3.  IF FPA NOT PRESENT SKIP TEST
0025  6733 ;       4.  CREATE A TEST LOOP OF 2
0025  6734 ;       5.  MODIFY DCS MICROCODE ACCORDING TO LOOP COUNT
0025  6735 ;       6.  EXECUTE DCS PROGRAM
0025  6736 ;               a.  DISABLE ^P INTERRUPTS
0025  6737 ;               b.  LOAD PC WITH ZERO TO CAUSE PREFETCH
0025  6738 ;               c.  DECODE FPA INSTRUCTION
0025  6739 ;               d.  PASS DATA TO THE FPA VIA MBUS
0025  6740 ;               e.  WAIT FOR FPA TO FINISH
0025  6741 ;               f.  READ FPA'S COPY OF CONDITION CODES TO RDM
0025  6742 ;       7.  TEST FOR CORRECT CONDITION CODES
0025  6743 ;       8.  IF NO ERROR GO TO STEP 5 FOR LAST LOOP
0025  6744 ;       9.  GO TO NEXT TEST
0025  6745 ;
0025  6746 ;
0025  6747 ; TEST PATTERNS:
0025  6748 ;
0025  6749 ;       ITERATION       MBUS DATA
0025  6750 ;           1           00000000
0025  6751 ;                       00000000
0025  6752 ;
0025  6753 ;           2           0000C080
0025  6754 ;                       00000000
0025  6755 ;
0025  6756 ;
0025  6757 ; LOGIC DESCRIPTION:
0025  6758 ;
0025  6759 ;       THIS TEST CHECKS THAT THE CPU CAN READ THE CONDITION CODES THAT ARE
0025  6760 ;       SET IN THE FPA.  THESE BITS ARE PASSED TO THE CPU VIA THE WBUS.
0025  6761 ;
0025  6762 ;
0025  6763 ; ASSUMPTIONS:
0025  6764 ;
0025  6765 ;       THIS TEST ASSUMES THE WBUS IS OPERATIONAL.
0025  6766 ;
0025  6767 ;
0025  6768 ; ERROR DESCRIPTION:
0025  6769 ;
0025  6770 ;       EXPECTED CONDITION CODES
0025  6771 ;       RECEIVED CONDITION CODES
0025  6772 ;       LOOP COUNT I
0025  6773 ;
0025  6774 ;--
```

```
                                0025  6775 ;*****************************************************************************
                                0025  6776
                                0025  6777
                                0025  6778         INITIALIZE                                      ;INIT THE CPU
                                0027  6779         FETCH           <^X0B>                          ;START DCS PROGRAM
                                002C  6780         BRSTCLK         <^X0D>                          ;END DCS PROGRAM
                                0030  6781         CMPREGMSK       CSTRP,1$,,1$,,W,NE              ;IS FPA PRESENT?
                                003C  6782         SKIP            10$,NOERROR                     ;SKIP IF PRESENT
                                0043  6783         PRINT_S         2$                              ;PRINT NO FPA
                                0047  6784         SKIP                                            ;GO TO NEXT TEST
                                004E  6785 10$:     LOOP            I,1,2                           ;TEST LOOP OF 3
                                0055  6786         LOADDCS         3$,I,06,2                       ;MODIFY DCS MICRO CODE
                                005C  6787         ERRLOOP                                         ;ERROR LOOP POINT
                                005E  6788         FETCH           0                               ;START DCS PROGRAM
                                0063  6789         BRSTCLK         <^X0A>                          ;END DCS PROGRAM
                                0067  6790         CMPREGMSK       WHREG,4$,I,5$,,L,               ;TEST CONDITION CODES
                                0073  6791         IFERROR         ,,<<FPA><MIC>>                  ;CODES INCORRECT
                                007A  6792         ENDLOOP         I                               ;END TEST LOOP
                                007D  6793         SKIP                                            ;GO TO NEXT TEST
                                0084  6794
                                0084  6795
                                0084  6796 BEGIN_TEST_DATA
                                0084
                                0084        ;-------------------------------------------------------------------------
                                0084  6797
                           0002 0084  6798 1$:      .WORD   ^X0002                                  ;FPA PRESENT
50 49 4B 53 20 54 53 45 54 0A 0D 00' 0086  6799 2$:      .ASCIC  <13><10>/TEST SKIPPED; FPA NOT PRESENT/<13><10>
54 4F 4E 20 41 50 46 20 3B 44 45 50  0092
      0A 0D 54 4E 45 53 45 52 50 20  009E
                                21   0086
                                     00A8  6800 3$:
U 06, 7700,4800,0364,0300,0471,1807  00A8  6801 ;% 06    FPA_M[TEMP0]                                ;I = 1
U 07, 7700,4800,0364,0300,0471,1808  00B3  6805 ;% 07    FPA_M[TEMP0]                                ;...
U 06, 7700,0801,0364,0300,0471,1807  00BE  6809 ;% 06    FPA_M[TEMP1]                                ;I = 2
U 07, 7700,4800,0364,0300,0471,1808  00C9  6813 ;% 07    FPA_M[TEMP0]                                ;...
                          00000004   00D4  6817 4$:      .LONG   ^X00000004                          ;RESULTS TABLE
                          00000008   00D8  6818          .LONG   ^X00000008                          ;...
                          0000000F   00DC  6819 5$:      .LONG   ^X0000000F                          ;MASK FOR CMPREGMSK PSEUDO OP
                                     00E0  6820
                                     00E0  6821
                                     00E0  6822 BEGIN_CPU_DATA
                                     0002  6823
                                     0002  6824 DCS_DATA
                                     0001  6825
U 00, 7700,C800,0364,0300,0470,1801  0001  6826 ;% 00    NOP
U 01, 7700,9800,D370,0360,2470,1802  000C  6830 ;% 01    CRAR_ZLIT16[0C0]                            ;DISABLE ^P INTERRUPTS
U 02, 7700,D800,D370,0320,2C70,1803  0017  6834 ;% 02    CONREGS_ZLIT16[040]                         ;...
U 03, 7700,4800,5BE4,03D8,4870,1804  0022  6838 ;% 03    PC_R[ZERO]                                  ;CAUSE PREFETCH
U 04, 7700,4800,0364,1700,0470,1805  002D  6842 ;% 04    IRD1TEST                                    ;START INSTRUCTION DECODE
U 05, 7700,C800,0364,0300,0471,1806  0038  6846 ;% 05    FPA_M[TEMP0]                                ;TICK FPA OUT OF INIT STATE
U 06, 7700,4800,0364,0300,0471,1807  0043  6850 ;% 06    FPA_M[TEMP0]                                ;PASS DATA TO FPA
U 07, 7700,4800,0364,0300,0471,1808  004E  6854 ;% 07    FPA_M[TEMP0]                                ;...
U 08, 7700,6800,0364,0300,0474,1809  0059  6858 ;% 08    WB_FPA,FPAWAIT                              ;WAIT FOR FPA TO FINISH
U 09, 6700,4800,0364,0300,0475,180A  0064  6862 ;% 09    RDM_WB,FPA/WBUS_FPA.CC                      ;READ FPA CONDITION CODES
U 0A, 7300,C800,0364,0300,0470,180B  006F  6866 ;% 0A    NOP,STOP.CLOCK                              ;STOP THE CPU CLOCK
U 0B, 7700,4800,0364,0300,0470,180C  007A  6870 ;% 0B    NOP
U 0C, 7700,4800,0364,C300,0470,1800  0085  6874 ;% 0C    BUT/NO.FPA,NEXT/1800                        ;TEST FPA STATUS
```

```
U 0D, 7300,C800,0364,0300,0470,180E  0090  6878 ;% 0D   NOP,STOP.CLOCK                    ;STOP THE CPU CLOCK
                                     009B  6882
                                     009B  6883 RTEMP_DATA
                                     0001  6884
                                00000000  0001  6885         .LONG   ^X00000000          ;DATA FOR FPA
                                0000C080  0005  6886         .LONG   ^X0000C080
                                     0009  6887
                                     0009  6888 CACHE_DATA
                                     0001  6889
                                00000076  0001  6890         .LONG   ^X00000076          ;CVTDF INSTRUCTION
                                00000000  0005  6891         .LONG   ^X00000000
                                     0009  6892
                                     0009  6893 END_CPU_DATA
                                     00E0  6894
                                     00E0  6895
                                     00E0  6896 END_TEST_DATA
                                     00E0
                                     00E0       ;-------------------------------------------------------------------------
                                     00E0  6897 ENDTEST
```

```
                        001C        .SBTTL  "TEST   0E8     TEST FPA COPY OF FU BIT
                        001C  6899 ;*******************************************************************************
                        001C  6900 ;++
                        001C  6901 ;
                        001C  6902 ; FUNCTIONAL DESCRIPTION:
                        001C  6903 ;
                        001C  6904 ;       THIS TEST VERIFIES THE FU PSL BIT CAN BE SET IN THE FPA
                        001C  6905 ;
                        001C  6906 ;
                        001C  6907 ; TEST ALGORITHM:
                        001C  6908 ;
                        001C  6909 ;       1.  EXECUTE DCS PROGRAM
                        001C  6910 ;               a.  BRANCH ON FPA PRESENT
                        001C  6911 ;       2.  TEST FOR FPA PRESENT
                        001C  6912 ;       3.  IF FPA NOT PRESENT SKIP THIS TEST
                        001C  6913 ;       4.  EXECUTE DCS PROGRAM
                        001C  6914 ;               a.  DISABLE ^P INTERRUPT
                        001C  6915 ;               b.  TICK FPA OUT OF IT'S INIT STATE
                        001C  6916 ;               c.  SET FPA FU BIT
                        001C  6917 ;               d.  LOAD PC WITH ZERO TO CAUSE PREFETCH
                        001C  6918 ;               e.  DECODE THE FPA INSTRUCTION
                        001C  6919 ;               f.  TICK FPA OUT OF INIT STATE
                        001C  6920 ;               g.  PASS DATA TO THE FPA
                        001C  6921 ;               h.  WAIT FOR FPA TO PROCESS INSTRUCTION
                        001C  6922 ;       5.  TEST CS ADDRESS FOR A FPA RESERVED OPND MICRO TRAP
                        001C  6923 ;       6.  IF NO ERROR GO TO NEXT TEST
                        001C  6924 ;
                        001C  6925 ;
                        001C  6926 ; TEST PATTERNS:
                        001C  6927 ;
                        001C  6928 ;       NONE
                        001C  6929 ;
                        001C  6930 ;
                        001C  6931 ; LOGIC DESCRIPTION:
                        001C  6932 ;
                        001C  6933 ;       THIS TEST ENSURES THE FPA CAN GENERATE A MICRO TRAP WHEN A FLOATING
                        001C  6934 ;       UNDERFLOW CONDITION OCCURS.
                        001C  6935 ;
                        001C  6936 ;
                        001C  6937 ; ASSUMPTIONS:
                        001C  6938 ;
                        001C  6939 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
                        001C  6940 ;
                        001C  6941 ;
                        001C  6942 ; ERROR DESCRIPTION:
                        001C  6943 ;
                        001C  6944 ;       EXPECTED CS ADDRESS
                        001C  6945 ;       RECEIVED CS ADDRESS
                        001C  6946 ;
                        001C  6947 ;--
                        001C  6948 ;*******************************************************************************
                        001C  6949
                        001C  6950
                        001C  6951         INITIALIZE                                      ;INIT THE CPU
                        001E  6952         FETCH           <^X0C>                          ;START DCS PROGRAM
                        0023  6953         BRSTCLK         <^X0E>                          ;END DCS PROGRAM
                        0027  6954         CMPREGMSK       CSTRP,1$,,1$,,W,NE              ;IS FPA PRESENT?
```

```
                                  0033          SKIP            10$,NOERROR                 ;SKIP IF PRESENT
                                  003A          PRINT_S         2$                          ;PRINT NO FPA
                                  003E          SKIP                                        ;GO TO NEXT TEST
                                  0045  10$:    EXPUTRAP                                    ;EXPECT A MICRO TRAP
                                  0047          ERRLOOP                                     ;ERROR LOOP POINT
                                  0049          FETCH           0                           ;START DCS PROGRAM
                                  004E          BRSTCLK         <^X0B>                      ;END DCS PROGRAM
                                  0052          CMPREGMSK       CSTRP,3$,,4$,,W,            ;TEST FOR MICRO TRAP ADDRESS
                                  005E          IFERROR         ,,<<FPA><MIC>>              ;INCORRECT ADDRESS
                                  0065          SKIP                                        ;GO TO NEXT TEST
                                  006C
                                  006C
                                  006C  BEGIN_TEST_DATA
                                  006C
                                  006C      ;-----------------------------------------------------------------------
                                  006C
                            0002  006C  1$:     .WORD   ^X0002                              ;FPA PRESENT BIT
50 49 4B 53 20 54 53 45 54 0A 0D 00' 006E  2$:     .ASCIC  <13><10>/TEST SKIPPED; FPA NOT PRESENT/<13><10>
54 4F 4E 20 41 50 46 20 3B 44 45 50  007A
      0A 0D 54 4E 45 53 45 52 50 20  0086
                              21  006E
                            002C  0090  3$:     .WORD   ^X002C                              ;FPA RESERVED OPERAND TRAP
                            3FFF  0092  4$:     .WORD   ^X3FFF                              ;MASK FOR CMPREGMSK PSEUDO OP
                                  0094
                                  0094  BEGIN_CPU_DATA
                                  0002
                                  0002  DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 7700,9800,D370,0360,2470,1802  000C  ;% 01   CRAR_ZLIT16[0C0]                           ;DISABLE ^P INTERRUPTS
U 02, 7700,D800,D370,0320,2C70,1803  0017  ;% 02   CONREGS_ZLIT16[040]                        ;...
U 03, 7700,4800,0364,0300,0471,1804  0022  ;% 03   FPA_M[TEMP0]                               ;TICK FPA OUT OF INIT STATE
U 04, 7700,8800,5BF4,0300,0070,1805  002D  ;% 04   PSL_R[TEMP0]                               ;SET FU BIT IN PSL
U 05, 7700,C800,5BE4,03D8,4870,1806  0038  ;% 05   PC_R[ZERO]                                 ;CAUSE PREFETCH
U 06, 7700,C800,0364,1700,0470,1807  0043  ;% 06   IRD1TEST                                   ;START INSTRUCTION DECODE
U 07, 7700,4800,0364,0300,0471,1808  004E  ;% 07   FPA_M[TEMP0]                               ;TICK FPA OUT OF INIT STATE
U 08, 7700,8801,0364,0300,0471,1809  0059  ;% 08   FPA_M[TEMP1]                               ;PASS DATA TO FPA
U 09, 7700,8801,0364,0300,0471,180A  0064  ;% 09   FPA_M[TEMP1]                               ;...
U 0A, 7700,E800,0364,0300,0474,180B  006F  ;% 0A   WB_FPA,FPAWAIT                             ;WAIT FOR FPA TO FINISH
U 0B, 7300,4800,0364,0300,0470,180C  007A  ;% 0B   NOP,STOP.CLOCK                             ;STOP THE CPU CLOCK
U 0C, 7700,C800,0364,0300,0470,180D  0085  ;% 0C   NOP
U 0D, 7700,4800,0364,C300,0470,1800  0090  ;% 0D   BUT/NO.FPA,NEXT/1800                       ;TEST FPA STATUS
U 0E, 7300,4800,0364,0300,0470,180F  009B  ;% 0E   NOP,STOP.CLOCK                             ;STOP THE CPU CLOCK
                                  00A6
                                  00A6  RTEMP_DATA
                                  0001
                        00000040  0001          .LONG   ^X00000040                          ;SET FU BIT IN PSL
                        00000080  0005          .LONG   ^X00000080                          ;DATA FOR FPA
                                  0009
                                  0009  CACHE_DATA
                                  0001
                        00000044  0001          .LONG   ^X00000044                          ;MULF2 INSTRUCTION
                        00000000  0005          .LONG   ^X00000000
                                  0009
                                  0009  END_CPU_DATA
                                  0094
                                  0094
```

```
0094  7052 END_TEST_DATA
0094
0094       ;-----------------------------------------------------------------------
0094  7053 ENDTEST
```

```
                0002  7056 .END
```

```
$CONTROL_C_CODE= 0000000A          CPU_RUN         = 00000010          END_TEST_226     000000B3 R     77
$FILE_NOT_FOUND= 00000009          CSAMR           = 00007404          END_TEST_227     000000C6 R     7D
$STN            = 00000000         CSBUF           = 00007424          END_TEST_228     000000ED R     83
$TEST_NUMBER    = 000000E9         CSTRP           = 00007422          END_TEST_229     000000F2 R     89
ACV             = 0000000F         CS_ADD_BUFF_HGH= 00007425           END_TEST_230     000000D3 R     8F
AC_LOW          = 00000080         CS_ADD_BUFF_LOW= 00007424           END_TEST_231     000000E0 R     95
ADD             = 00000019         CS_ADD_TRAP_HGH= 00007423           END_TEST_232     00000094 R     9B
ADDR_MATCH_HI   = 00007405         CS_ADD_TRAP_LOW= 00007422           EN_TRACE_REG    = 00000008
ADDR_MATCH_LO   = 00007404         CYCLE_FLAG      = 00000008          ERRLOG_FLAG     = 00000000
ADK             = 0000000B         CYCLE_KEY       = 00000020          ERROR_FLAG      = 00000010
ALK             = 00000000         D0              = 00000008          ERROR_LOG_ADR   = 000032FC
ALP             = 00000016         D1              = 0000000E          ERR_LOG_INIT    = 00000001
ARG_CNT         = 0000005F R   9B  D2              = 00000000          EXP_STALL_FLAG  = 00000010
A_REG_BYTE_0    = 00007410         DATA_FLAG       = 00000000          EXP_UTRAP_FLAG  = 00000040
A_REG_BYTE_1    = 00007411         DATA_LENGTH     = 00000002          FAC             = 00000020
A_REG_BYTE_2    = 00007412         DCSAD           = 00007401          FAULT           = 00000002
A_REG_BYTE_3    = 00007413         DCSCF           = 00007403          FCC             = 00000015
B               = 00000001         DCSCR           = 00007402          FCS             = 00000021
BADD1           = 00007426         DCSDA           = 00007400          FETCH_FLAG      = 00000008
BELL_FLAG       = 00000008         DCS_ADDR_CLEAR  = 00000002          FEX             = 0000001E
BELL_KEY        = 0000000E         DCS_ADDR_REG_RO= 00007427           FFH             = 00000024
BUFF_ADDR_LOW   = 00007426         DCS_ADDR_REG_WO= 00007401           FFL             = 00000023
BURST_STOP_FLAG= 00000080          DCS_CTRL_FILE   = 00007403          FIC             = 0000001F
BYTE            = 00000001         DCS_CTRL_REG    = 00007402          FIO             = 0000001D
CACHE_FLAG      = 00000000         DCS_DATA_REG    = 00007400          FLAG_KEY        = 00000004
CACHE_FLAG1     = 00000000         DCS_FLAG        = 00000000          FMR             = 00000022
CACHE_HEAD      = 00000000 R   9E  DCS_FLAG1       = 00000000          FPA             = 00000002
CACHE_LENGTH    = 00000001         DCS_HEAD        = 00000000 R   9C   FP_BOOT_0       = 00000001
CAK             = 0000000A         DCS_LENGTH      = 00000001          FP_BOOT_1       = 00000002
CCC             = 00000004         DCS_SCRATCH_ADR= 00000036           FP_START_0      = 00000004
CCS             = 00000005         DCS_START       = 00001800          FP_START_1      = 00000008
CF_KEY          = 00000024         DC_LOW          = 00000040          FQA             = 00000014
CLA             = 00000003         DIAGNOSE_FLAG   = 00000004          FRONT1          = 00007420
CLEAR_CTRL_FILE= 00000002          DPM             = 00000000          FRONT2          = 00007421
CLKDCS          = 00007427         DUM1            = FFFFFFF8          FRONT_PNL_1     = 00007420
CLK_CTRL_0      = 00000020         ENDTEST_LOC     = 00000099 R   9B   FRONT_PNL_2     = 00007421
CLK_CTRL_0_RO   = 00000040         END_SLICE       = 00000025          FRONT_PNL_LOCK  = 00000080
CLK_CTRL_1      = 00000040         END_TEST_206     00000108 R   02    HALT_FLAG       = 00000001
CLK_CTRL_1_RO   = 00000080         END_TEST_207     000000A4 R   09    HALT_KEY        = 00000008
CLOCK_STOPED    = 00000080         END_TEST_208     000000AF R   0F    HALT_ON_MATCH   = 00000001
CMI             = 00000007         END_TEST_209     000001BB R   15    HEAD            = FFFF7C00
CMICTL          = 0000741C         END_TEST_210     000000CF R   1B    HIGH            = 00000001
CMI_COMPLETE    = 00000020         END_TEST_211     000000BE R   21    H_FROM_WBUS     = 00000010
CMI_CTRL_CLEAR = 00000001          END_TEST_212     000000F7 R   27    IB_FLAG         = 00000020
CMI_CTRL_REG    = 0000741C         END_TEST_213     00000118 R   2D    IB_KEY          = 00000012
CMI_GO          = 00000008         END_TEST_214     00000489 R   33    INSTR_FLAG      = 00000004
CMI_READ        = 00000040         END_TEST_215     000000F9 R   39    INSTR_KEY       = 00000018
CMI_STATUS_0    = 00000008         END_TEST_216     0000048F R   3F    INT             = 00000010
CMI_STATUS_1    = 00000010         END_TEST_217     00000107 R   45    IRD             = 00000005
CMI_WRITE       = 00000020         END_TEST_218     0000047B R   4B    I_INDEX_FLAG    = 00000000
CML             = 0000000E         END_TEST_219     0000018B R   51    I_INIT          = 00000000
COMPOSIT_LENGTH= 00000004          END_TEST_220     000000AC R   57    J_INDEX_FLAG    = 00000000
CON             = 00000011         END_TEST_221     00000071 R   59    J_INIT          = 00000000
CONTINUE_FLAG   = 00000002         END_TEST_222     0000017E R   5F    K_INDEX_FLAG    = 00000000
CONTROL_C_FLAG = 00000001          END_TEST_223     0000009F R   65    K_INIT          = 00000000
CONT_FLAG       = 00000020         END_TEST_224     000000EF R   6B    L               = 00000004
CONT_KEY        = 0000001C         END_TEST_225     000000D0 R   71    LIST_END        = 00000065 R   9B
```

```
LIST_HEAD       = 00000060 R     9B      NER_KEY         = 0000000C              RTEMP_FLAG1     = 00000000
LIST_LENGTH     = 00000005               NOERROR         = 00000001              RTEMP_HEAD      = 00000000 R     9D
LITERAL         = 00000000               NOTEQUAL        = 00000003              RTEMP_LENGTH    = 00000001
LONG            = 00000004               ONEQUAL         = 00000002              RT_KEY          = 0000002C
LOOP_ERROR_FLAG= 00000020                ONERROR         = 00000000              SAC             = 00000009
LOOP_FLAG       = 00000002               OP_BEGIN_LOOP   = 00000008              SA_CLOCK        = 00000080
LOOP_KEY        = 0000000A               OP_BURST_CLOCK  = 00000002              SA_FLAG         = 00000010
LOST_FLAG       = 00000010               OP_COMPARE_REG  = 00000004              SA_KEY          = 00000010
LOST_KEY        = 0000001A               OP_COMPARE_REGM= 00000006               SA_START_STOP   = 00000080
LOW             = 00000000               OP_COMPARE_VB   = 00000020              SBE_FLAG        = 00000020
M               = 0000000A               OP_DUMPLOG      = 00000030              SBE_KEY         = 00000042
M$TEST_SIZE     = 000007A2               OP_ENABL_STALL  = 00000038              SECTION_FLAG    = 00000000
M$TEST_SIZE_D   = 000003E2               OP_ENABL_TRAP   = 0000002E              SF_KEY          = 00000040
MA0             = 00000008               OP_END_FILE     = 00000024              SOMM_FLAG       = 00000001
MA1             = 0000000A               OP_END_LOOP     = 0000000A              SOMM_KEY        = 00000006
MA2             = 0000000C               OP_END_TEST     = 0000000C              SPA             = 00000002
MAD             = 00000013               OP_ERRLOG       = 00000032              SRK             = 00000001
MAIN32          = 0000741D               OP_ERROR_LOOP   = 0000000E              SRM             = 00000017
MAINT_A_TO_CMI  = 00000080               OP_FETCH        = 00000022              SR_KEY          = 0000003C
MAINT_CMI_TO_MH= 00000020                OP_IF_ERROR     = 00000012              STACK_SIZE      = 00000400
MAINT_DCS_ENABL= 00000004                OP_INC_INDEX    = 0000002A              STATUS          = 00007406
MAINT_MD_TO_CMI= 00000040                OP_INIT         = 00000014              STATUS_REG      = 00007406
MAINT_REG       = 0000741D               OP_LOAD_DCS     = 00000000              STEP_KEY        = 0000001E
MAINT_STB_INH   = 00000002               OP_LOAD_FIELD   = 00000026              STOP_CLOCK      = 00000004
MAINT_TRAP_OFF  = 00000001               OP_LOAD_INDEX   = 00000028              STROBE_CMI      = 00000040
MAINT_WB_TO_WH  = 00000008               OP_LOAD_REG     = 00000016              ST_KEY          = 0000001E
MAINT_WD_TO_WB  = 00000010               OP_MASK         = 00000018              TEMP            = 0000000E
MAP             = 00000012               OP_NEW_TEST     = 0000001A              TEMP1           = 000000F2
MAREG           = 00007410               OP_PATTERN_GEN  = 00000010              TEMP2           = 000000B8
MASTER_HALT_EN  = 00000008               OP_PRINT_N      = 00000036              TEMP3           = 00000002
MAX_ARGUMENTS   = 00000010               OP_PRINT_S      = 00000034              TEMP4           = 00000002 R     9A
MA_KEY          = 00000038               OP_SAVE_INDEX   = 0000002C              TEST            = 00000004
MC0             = 00000003               OP_SKIP         = 0000001C              TEST_FLAG       = 00000000
MC1             = 00000009               OP_SUB_TEST     = 0000001E              TEST_HEAD       = 00000019 R     9B
MC2             = 0000000B               OVERLAY_HEAD    = 00000000 R     A0     TEST_KEY        = 00000000
MDL             = 0000001B               PARITY_ERROR    = 00000020              TEST_LENGTH     = 00000002
MDR             = 00000018               PAR_CHK_ENABL   = 00000008              TEST_SECT_HEAD  = 00000000 R     9B
MDREG           = 00007414               PAR_STOP_ENABL  = 00000010              THIS_TEST       = 000000E8
MD_KEY          = 00000034               PASS_KEY        = 00000002              TICK_FLAG       = 00000002
MD_REG_BYTE_0   = 00007414               PB_INIT         = 00000040              TICK_KEY        = 00000022
MD_REG_BYTE_1   = 00007415               PB_KEY          = 00000036              TOK             = 00000006
MD_REG_BYTE_2   = 00007416               PC_KEY          = 00000030              TRACE_ENABL     = 00000004
MD_REG_BYTE_3   = 00007417               PHB             = 00000007              TRAP_HALT_EN    = 00000020
MEC             = 0000001A               PRK             = 0000000C              TRAP_OFF        = 00000002
MHREG           = 00007418               PS_KEY          = 0000003E              TR_FLAG         = 00000080
MH_REG_BYTE_0   = 00007418               QA_FLAG         = 00000040              TR_KEY          = 00000016
MH_REG_BYTE_1   = 00007419               QA_KEY          = 00000014              TSTSPAN_FLAG    = 00000040
MH_REG_BYTE_2   = 0000741A               QV_FLAG         = 00000002              UBI             = 00000006
MH_REG_BYTE_3   = 0000741B               QV_KEY          = 00000028              UDP             = 0000001C
MIC             = 00000001               RDCTRL          = 0000741E              UTR             = 0000000D
MICROWORD       = 0000000A               RDM             = 00000004              VA_KEY          = 00000032
MICRO_ADDR_INH  = 00000080               RDM_FLAG        = 00000004              VBUS_CLOCK      = 00000080
MONITOR_SIZE    = 00002C5E               RDM_KEY         = 0000002A              VBUS_COMPARE    = 00000080
MSQ             = 00000008               RDM_LAST        = 00000008              VBUS_LOAD       = 00000010
MT_KEY          = 0000002E               RD_CTRL_REG     = 0000741E              VBUS_SERIAL_IN  = 00000040
N               = 00000002               REMOTE          = 00000001              VBUS_SERIAL_OUT= 00000001
NER_FLAG        = 00000004               RTEMP_FLAG      = 00000000              VB_KEY          = 00000026
```

```
V_BUS_STROBE    = 00000001
W               = 00000002
WBUS_FROM_D     = 00000020
WDREG           = 0000742C
WD_KEY          = 0000003A
WD_REG_BYTE_0   = 0000742C
WD_REG_BYTE_1   = 0000742D
WD_REG_BYTE_2   = 0000742E
WD_REG_BYTE_3   = 0000742F
WHREG           = 00007430
WH_REG_BYTE_0   = 00007430
WH_REG_BYTE_1   = 00007431
WH_REG_BYTE_2   = 00007432
WH_REG_BYTE_3   = 00007433
WORD            = 00000002
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
X_0206_D	00000002 (2.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0206_T	00000180 (384.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0206_GDCS	0000018D (397.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0206_ERTEMP	00000021 (33.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0206_FCACHE	00000001 (1.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0206_HFILL	0000004F (79.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
DIRECTORY	00000029 (41.)	07 (7.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0207_D	00000002 (2.)	08 (8.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0207_T	00000100 (256.)	09 (9.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0207_GDCS	0000014B (331.)	0A (10.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0207_ERTEMP	00000015 (21.)	0B (11.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0207_FCACHE	00000001 (1.)	0C (12.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0207_HFILL	0000001D (29.)	0D (13.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0208_D	00000002 (2.)	0E (14.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0208_T	00000100 (256.)	0F (15.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0208_GDCS	0000014B (331.)	10 (16.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0208_ERTEMP	00000015 (21.)	11 (17.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0208_FCACHE	00000001 (1.)	12 (18.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0208_HFILL	0000001D (29.)	13 (19.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0209_D	00000002 (2.)	14 (20.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0209_T	00000200 (512.)	15 (21.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0209_GDCS	0000014B (331.)	16 (22.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0209_ERTEMP	00000015 (21.)	17 (23.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0209_FCACHE	00000001 (1.)	18 (24.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0209_HFILL	0000001D (29.)	19 (25.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0210_D	00000002 (2.)	1A (26.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0210_T	00000100 (256.)	1B (27.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0210_GDCS	0000014B (331.)	1C (28.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0210_ERTEMP	00000015 (21.)	1D (29.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0210_FCACHE	00000001 (1.)	1E (30.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0210_HFILL	0000001D (29.)	1F (31.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0211_D	00000002 (2.)	20 (32.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0211_T	00000100 (256.)	21 (33.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0211_GDCS	00000114 (276.)	22 (34.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0211_ERTEMP	00000005 (5.)	23 (35.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0211_FCACHE	00000001 (1.)	24 (36.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0211_HFILL	00000064 (100.)	25 (37.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0212_D	00000002 (2.)	26 (38.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0212_T	00000100 (256.)	27 (39.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0212_GDCS	0000007A (122.)	28 (40.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0212_ERTEMP	0000000D (13.)	29 (41.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0212_FCACHE	000000C9 (201.)	2A (42.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0212_HFILL	0000002E (46.)	2B (43.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0213_D	00000002 (2.)	2C (44.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0213_T	00000180 (384.)	2D (45.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0213_GDCS	0000007A (122.)	2E (46.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0213_ERTEMP	0000000D (13.)	2F (47.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0213_FCACHE	000000F5 (245.)	30 (48.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
X_0213_HFILL	00000002 (2.)	31 (49.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

X_0214_D	00000002	(2.)	32	(50.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0214_T	00000500	(1280.)	33	(51.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0214_GDCS	000000D2	(210.)	34	(52.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0214_ERTEMP	00000019	(25.)	35	(53.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0214_FCACHE	00000001	(1.)	36	(54.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0214_HFILL	00000012	(18.)	37	(55.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0215_D	00000002	(2.)	38	(56.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0215_T	00000100	(256.)	39	(57.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0215_GDCS	000000D2	(210.)	3A	(58.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0215_ERTEMP	00000019	(25.)	3B	(59.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0215_FCACHE	00000001	(1.)	3C	(60.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0215_HFILL	00000012	(18.)	3D	(61.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0216_D	00000002	(2.)	3E	(62.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0216_T	00000500	(1280.)	3F	(63.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0216_GDCS	00000114	(276.)	40	(64.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0216_ERTEMP	00000019	(25.)	41	(65.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0216_FCACHE	00000001	(1.)	42	(66.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0216_HFILL	00000050	(80.)	43	(67.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0217_D	00000002	(2.)	44	(68.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0217_T	00000180	(384.)	45	(69.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0217_GDCS	00000114	(276.)	46	(70.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0217_ERTEMP	00000019	(25.)	47	(71.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0217_FCACHE	00000001	(1.)	48	(72.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0217_HFILL	00000050	(80.)	49	(73.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0218_D	00000002	(2.)	4A	(74.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0218_T	00000500	(1280.)	4B	(75.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0218_GDCS	0000011F	(287.)	4C	(76.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0218_ERTEMP	00000019	(25.)	4D	(77.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0218_FCACHE	00000001	(1.)	4E	(78.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0218_HFILL	00000045	(69.)	4F	(79.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0219_D	00000002	(2.)	50	(80.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0219_T	00000200	(512.)	51	(81.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0219_GDCS	0000011F	(287.)	52	(82.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0219_ERTEMP	00000019	(25.)	53	(83.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0219_FCACHE	00000001	(1.)	54	(84.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0219_HFILL	00000045	(69.)	55	(85.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0220_D	00000002	(2.)	56	(86.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0220_T	000000FE	(254.)	57	(87.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0221_D	00000002	(2.)	58	(88.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0221_T	00000080	(128.)	59	(89.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0221_GDCS	00000038	(56.)	5A	(90.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0221_ERTEMP	00000021	(33.)	5B	(91.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0221_FCACHE	00000001	(1.)	5C	(92.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0221_HFILL	00000024	(36.)	5D	(93.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0222_D	00000002	(2.)	5E	(94.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0222_T	00000200	(512.)	5F	(95.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0222_ERTEMP	00000005	(5.)	60	(96.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0222_FCACHE	00000001	(1.)	61	(97.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0222_GDCS	00000001	(1.)	62	(98.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0222_HFILL	00000077	(119.)	63	(99.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0223_D	00000002	(2.)	64	(100.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0223_T	00000100	(256.)	65	(101.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0223_GDCS	0000006F	(111.)	66	(102.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0223_FCACHE	00000009	(9.)	67	(103.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0223_ERTEMP	00000001	(1.)	68	(104.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0223_HFILL	00000005	(5.)	69	(105.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE
X_0224_D	00000002	(2.)	6A	(106.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT NOVEC BYTE

X_0224_T	00000100	(256.)	6B	(107.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0224_GDCS	00000090	(144.)	6C	(108.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0224_FCACHE	00000009	(9.)	6D	(109.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0224_ERTEMP	00000001	(1.)	6E	(110.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0224_HFILL	00000064	(100.)	6F	(111.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0225_D	00000002	(2.)	70	(112.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0225_T	00000100	(256.)	71	(113.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0225_GDCS	000000A6	(166.)	72	(114.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0225_ERTEMP	00000005	(5.)	73	(115.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0225_FCACHE	00000009	(9.)	74	(116.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0225_HFILL	0000004A	(74.)	75	(117.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0226_D	00000002	(2.)	76	(118.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0226_T	00000100	(256.)	77	(119.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0226_GDCS	000000B1	(177.)	78	(120.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0226_ERTEMP	00000009	(9.)	79	(121.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0226_FCACHE	00000009	(9.)	7A	(122.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0226_HFILL	0000003B	(59.)	7B	(123.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0227_D	00000002	(2.)	7C	(124.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0227_T	00000100	(256.)	7D	(125.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0227_GDCS	0000009B	(155.)	7E	(126.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0227_ERTEMP	00000009	(9.)	7F	(127.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0227_FCACHE	00000009	(9.)	80	(128.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0227_HFILL	00000051	(81.)	81	(129.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0228_D	00000002	(2.)	82	(130.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0228_T	00000100	(256.)	83	(131.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0228_GDCS	000000A6	(166.)	84	(132.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0228_ERTEMP	0000000D	(13.)	85	(133.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0228_FCACHE	00000009	(9.)	86	(134.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0228_HFILL	00000042	(66.)	87	(135.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0229_D	00000002	(2.)	88	(136.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0229_T	00000100	(256.)	89	(137.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0229_GDCS	000000BC	(188.)	8A	(138.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0229_ERTEMP	00000009	(9.)	8B	(139.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0229_FCACHE	00000009	(9.)	8C	(140.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0229_HFILL	00000030	(48.)	8D	(141.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0230_D	00000002	(2.)	8E	(142.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0230_T	00000100	(256.)	8F	(143.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0230_GDCS	000000B1	(177.)	90	(144.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0230_ERTEMP	00000011	(17.)	91	(145.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0230_FCACHE	00000009	(9.)	92	(146.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0230_HFILL	00000033	(51.)	93	(147.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0231_D	00000002	(2.)	94	(148.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0231_T	00000100	(256.)	95	(149.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0231_GDCS	0000009B	(155.)	96	(150.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0231_ERTEMP	00000009	(9.)	97	(151.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0231_FCACHE	00000009	(9.)	98	(152.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0231_HFILL	00000051	(81.)	99	(153.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0232_D	00000002	(2.)	9A	(154.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0232_T	00000100	(256.)	9B	(155.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0232_GDCS	000000A6	(166.)	9C	(156.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0232_ERTEMP	00000009	(9.)	9D	(157.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0232_FCACHE	00000009	(9.)	9E	(158.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0232_HFILL	00000046	(70.)	9F	(159.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
X_0233_D	00000000	(0.)	A0	(160.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
A_DIRECTORY	00000002	(2.)	A1	(161.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	24	00:00:00.19	00:00:00.66
Command processing	862	00:00:00.94	00:00:01.59
Pass 1	2932	00:01:32.70	00:01:51.23
Symbol table sort	0	00:00:00.57	00:00:00.65
Pass 2	3088	00:00:29.55	00:00:43.07
Symbol table output	32	00:00:00.22	00:00:00.22
Psect synopsis output	48	00:00:00.60	00:00:00.68
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	6987	00:02:04.79	00:02:38.11

The working set limit was 2048 pages.
1167893 bytes (2282 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 357 non-local and 153 local symbols.
7801 source lines were read in Pass 1, producing 347 object records in Pass 2.
104 pages of virtual memory were used to define 49 macros.

Macro library statistics

Macro library name	Macros defined
DISK$UCODPACK00:[VAX750]COMETMAC.MLB;1	49
SYS$COMMON:[SYSLIB]STARLET.MLB;2	0
TOTALS (all libraries)	49

2057 GETS were required to define 49 macros.

There were 0 errors, 3 warnings and 0 information messages, on lines:
 3102 (1) 3795 (1) 4560 (1)

MACRO UCOD$0:[VAX750.DPM]TITLE+DISK$UCODPACK00:[VAX750]COMETASM+UCOD$0:[VAX750.DPM]ECKAB6/LIS=ECKAB6/OBJ=ECKAB6